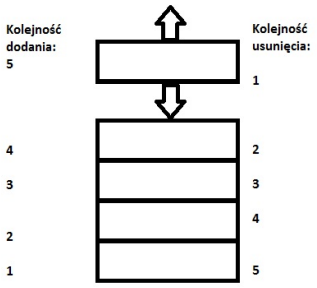
Stos i notacja postfix

13 października 2015

1 Stos

Stos (inaczej kolejka LIFO - last in, first out) jest abstrakcyjnym typem danych. Struktura ta charakteryzuje się tym, że dodawanie i usuwanie nowych danych zawsze odbywa się na tym samym końcu struktury.



Rysunek 1: Kolejność dodawania i usuwanie elementów na stosie

Na stosie wykonuje się następujące operacje:

- push(obiekt) dodanie nowego obiektu na szczyt stosu. Parametrem wejściowym jest obiekt, nic nie jest zwracane.
- pop() usunięcie obiektu ze szczytu stosu. Brak parametrów wejściowych, zwracany jest obiekt ze szczytu stosu. Stos jest modyfikowany.
- peek() zwraca szczytowy element stosu, ale go nie usuwa. Brak parametrów wejściowych, stos nie jest modyfikowany.
- isEmpty() zwraca wartość True jeżeli stos jest pusty, w przeciwnym wypadku wartość False.
- size() zwraca liczbę obiektów znajdującą się na stosie.

Pierwsza część zadania polega na iplementacji stosu, wraz z opisanymi wyżej operacjami.

2 Notacja postfix

2.1 Konwersja do notacji postfix

Druga część zadania polega na implementacji algorytmu konwertującego zapis działania matematycznego infix do zapisu postfix. W standardowym zapisie (infix) mamy np:

$$A + B * (C - D) \tag{1}$$

By wykonać w sposób prawidłowy te operacje musimy znać kolejność działań oraz sposób używania nawiasów. To samo wyrażenie można zapisać w inny sposób:

$$ABCD - *+ \tag{2}$$

Algorytm postępowania:

- Utwórz pusty stos na którym przechowywane będą operatory działania oraz pustą listę/tablicę wynikową

- Wejściową tablicę z danymi czyta się od lewej do prawej
  - jeżeli znak jest liczbą/(symbolem liczby tak jak A), należy dopisać go do wyjścia
  - jeżeli znak jest lewym nawiasem (, należy dodać go do stosy
  - jeżeli znak jest prawym nawiasem), należy ze stosu usunąć wszystkie elementy aż do znalezienia nawiasy lewego. Wszystkie elementy (poza nawiasami) dodaje się do wyjścia
  - jeżeli znak jest operatorem działania, należy
    - usunąć ze stosu wszystkie operatory o tym samym/wyższym pierwszeństwie działania
    - dodać aktualnie przeczytany operator do stosu
- jeżeli w wejściowej tablicy nie pozostały już żadne dane, to wszystkie obiekty ze stosu należy dopisać do wyjścia

Przykład prosty Wyrażenie wejściowe: $(A + B) * C$ Kroki algorytmu:

1. znak wejścia: (. Dopisanie do stosu. Wyjście: puste. Stos: (
2. znak wejścia A. Dopisanie do wyjścia. Wyjście: A. Stos: (
3. znak wejścia +. Dopisanie do stosu. Wyjście: A. Stos: (, +
4. znak wejścia B. Dopisanie do wyjścia. Wyjście: A B. Stos: (, +
5. znak wejścia). Usunięcie elementów ze stosu aż do znaku). Dopisanie ich do listy w odpowiedniej kolejności. Wyjście: A B + . Stos: pusty
6. znak wejścia *. Dopisanie do stosu. Wyjście: A B +. Stos: *
7. znak wejścia C. Dopisanie do wyjścia. Wyjście: A B + C. Stos: *
8. brak nowych elementów wejścia. Usunięcie ze stosu wszystkich znaków Wyjście: A B + C *. Stos: pusty.

Przykład bardziej złożony Wyrażenie wejściowe: $A * (B + C) - D/(E - F)$. Kroki algorytmu:

1. znak wejścia: A. Dopisanie do wyjścia. Wyjście: A. Stos: pusty.
2. znak wejścia *. Dopisanie do stosu. Wyjście: A. Stos: *
3. znak wejścia (. Dopisanie do stosu. Wyjście: A. Stos: * , (
4. znak wejścia B. Dopisanie do wyjścia. Wyjście: A B. Stos: * , (
5. znak wejścia +. Dopisanie do stosu. Wyjście: A B. Stos: * , (+
6. znak wejścia C. Dopisanie do wyjścia. Wyjście: A B C. Stos: * , (+
7. znak wejścia). Usunięcie elementów ze stosu aż do znaku). Dopisanie ich do listy w odpowiedniej kolejności. Wyjście: A B C +. Stos: *
8. znak wejścia -. Dopisanie do stosu. Ponieważ poprzedni element na stosie ma wyższy priorytet (* - mnożenie), przed dopisaniem do stosu -, do wyjścia dopisujemy * i usuwamy * ze stosu. Wyjście: A B C + *. Stos: -
9. znak wejścia D. Dopisanie do wyjścia. Wyjście: A B C + * D. Stos: -
10. znak wejścia /. Dopisanie do stosu (ponieważ poprzedni element to odejmowanie, które ma niższy priorytet od dzielenia, stosu nie trzeba modyfikować). Wyjście: A B C + * D. Stos: -, /
11. znak wejścia (. Dopisanie do stosu. Wyjście: A B C + * D. Stos: -, /, (
12. znak wejścia -. Dopisanie do stosu. Wyjście: A B C + * D. Stos: -, /, (, -
13. znak wejścia F. Dopisanie do wyjścia. Wyjście: A B C + * D E F. Stos: -, /, (, -
14. znak wejścia). Ususnięcie ze stosu wszystkich znaków aż do (. Wyjście: A B C + * D E F -. Stos: -, /,
15. brak nowych elementów wejścia. Usunięcie ze stosu wszystkich znaków Wyjście: A B C + * D E F - / -. Stos: pusty.

Inne przykałdy (bez rozpisania):
Wejście: $A * (B + C/(D - E))$
Wyjście: $ABCDE - / + *$
Wejście: $A * B + C/((D - E) * F)$
Wyjście: $AB * CDE - F * /+$

2.2 Obliczanie wyrażeń zapisanych w notacji postfix

W celu obliczenia wyrażenia zapisanego w notacji postfix należy posłużyć się następującym algorytmem:

- Utwórz pusty stos na którym przechowywane będą dodawane liczby
- Wejściową tablicę z danymi czyta się od lewej do prawej
  - jeżeli znak jest liczbą/(symbolem liczby tak jak A), należy dodać go do stosu
  - jeżeli znak jest operatorem działania, należy
    - usunąć ze stosu dwie ostatnio zapisane liczby
    - wykonać na nich działanie (w odpowiedniej kolejności, liczba głębiej na stosie jest pierwszą liczbą w działaniu: ma to znaczenie przy wykonywaniu odejmowania i dzielenia)
    - wynik działania odłożyć na stos
- jeżeli z listy wejściowej przeczytano wszystkie znaki, to do wynikiem działania jest liczba znajdująca się na stosie

Przykład (prosty) Wyrażenie wejściowe: 6,5,15,*,+ (w notacji infix 6+5*15) Kroki algorytmu:

1. znak wejścia: 6. Dopisanie do stosu. Stos: 6
2. znak wejścia 5. Dopisanie do stosu. Stos: 6,5
3. znak wejścia 15. Dopisanie do stosu. Stos: 6,5,15
4. znak wejścia *. Usunięcie ze stosu dwóch ostatnich elementów (15,5). Wykonanie działania 5*15 = 75. Odłożenie wyniku na stos. Stos 6,75
5. znak wejścia: +. Usunięcie ze stosu dwóch ostatnich elementów (75,6). Wykonanie działania 6+75 = 81. Odłożenie wyniku na stos. Stos 81
6. Ponieważ lista wejściowa nie zawiera więcej elementów to na stosie znajduje sie wynik działania (81)

Przykład (trudniejszy) Wyrażenie wejściowe: 6,5,-,8,4,-,*,4,3,-,/ (w notacji infix (6-5)*(8-4)/(4-3)) Kroki algorytmu:

1. znak wejścia: 6. Dopisanie do stosu. Stos: 6
2. znak wejścia 5. Dopisanie do stosu. Stos: 6,5
3. znak wejścia -. Usunięcie ze stosu dwóch ostatnich elementów (5,6). Wykonanie działania (uwaga na kolejność) 6 - 5 = 1. Dopisanie wyniku do stosu. Stos: 1
4. znak wejścia 8. Dopisanie do stosu. Stos: 1,8
5. znak wejścia: 4. Dopisanie do stosu. Stos: 1,8,4
6. znak wejścia -. Usunięcie ze stosu dwóch ostatnich elementów (4,8). Wykonanie działania (uwaga na kolejność) 8 - 4 = 4. Dopisanie wyniku do stosu. Stos: 1,4
7. znak wejścia *. Usunięcie ze stosu dwóch ostatnich elementów (4,1). Wykonanie działania 1 * 4 = 4. Dopisanie wyniku do stosu. Stos: 4
8. znak wejścia: 4. Dopisanie do stosu. Stos: 4,4
9. znak wejścia: 3. Dopisanie do stosu. Stos: 4,4,3
10. znak wejścia -. Usunięcie ze stosu dwóch ostatnich elementów (3,4). Wykonanie działania (uwaga na kolejność) 4 - 3 = 1. Dopisanie wyniku do stosu. Stos: 4,1
11. znak wejścia /. Usunięcie ze stosu dwóch ostatnich elementów (1,4). Wykonanie działania (uwaga na kolejność) 4 / 1 = 4. Dopisanie wyniku do stosu. Stos: 4
12. Ponieważ lista wejściowa nie zawiera więcej elementów to na stosie znajduje sie wynik działania (4)

2.3 Punktacja

- 4 punkty - implementacja stosu (za korzystanie ze zdefiniowanych struktur danych punkty nie będą przyznane)
- 3 punkty - implementacja algorytmu konwertującego do notacji postfix
- 3 punkty - implementacja algorytmu obliczającego wartość wyrażenia zapisanego w notacji postfix