Algorytmy i struktury danych

Laboratorium Stos implementacja tablicowa

Zad. 1 Zapoznaj się tablicową implementacją stosu *TArrayStack<E>* przedstawioną na wykładzie. Klasa posiada zaimplementowany konstruktor, oraz metody *top, pop, push, size, isEmpty oraz isFull*. Rozszerz implementację/zaimplementuj metody:

a) *String printStack()* pozwalającą na wyświetlenie stosu w postaci kolejnych elementów tablicy. Przykładowo jeżeli do stosu przekażemy napisy: „ene", „due", „rike", „fake", to metoda produkuje łańcuch znaków postaci:

```
Tab[3]=fake <- top
Tab[2]=rike
Tab[1]=due
Tab[0]=ene
```

b) *void extendMemory()* pozwalającą na rozszerzenie pamięci dla stosu. Metoda rozszerza pamięć o dodatkowe 50% obecnej. Sprawdź metodę *System.arraycopy()*.

c) *int deepLevel(item)* sprawdzającą ja głęboko na stosie jest dostępny szukany element *item*.

Zad. 2 Przy zastosowaniu algorytmu obliczania wyrażenie ONP przedstaw proces obliczania poniższych wyrażeń w postaci tabelki:

a) 100 50 - 10 / 25 10 % 2 + 4 * +
b) 22 5 / 23 6 % 5 * + 10 5 3 - / -
c) 3 5 + 3 % 30 12 - 10 * + 33 10 / +
d) 2 3 + 10 * 20 4 + 6 / - 13 2 + 7 % +
e) 5 4 + 3 * 13 5 % 7 3 / * + 13 -

Zad. 3 Przy zastosowaniu algorytmu przekształcania wyrażenia algebraicznego na wyrażenie ONP przedstaw proces zamiany w postaci tabelki:

a) (5 + 4) * 3 + (13 % 5) * (7 / 3) - 13
b) (2 + 3) * 10 - (20 + 4) / 6 + (13 + 2) % 7
c) (3 + 5) % 3 + (30 - 12) * 10 + 33 / 10
d) 22 / 5 + (23 % 6) * 5 - 10 / (5 - 3)
e) (100 - 50) / 10 + (25 % 10 + 2) * 4

Zad. 4 Zaprogramuj konwerter pozwalający na generowanie wyrażenia ONP na podstawie algebraicznego wyrażenia z możliwymi nawiasami.

Klasa RPNGenerator zawiera pole przechowujące stos znaków oraz konstruktor ustawiający rozmiar stosu. Klasa posiada metodę generującą wyrażenie ONP na podstawie wyrażenia algebraicznego przechowywanego w pliku którego nazwę przyjmuje ta metoda jako parametr. Metoda zwraca łańcuch znaków z wyrażeniem ONP. W razie potrzeby zaimplementuj prywatne metody pomocnicze.

Zad. 5 Zaprogramuj kalkulator wyrażeń ONP. Napisz klasę RPNCalculation, która posiada pole przechowujące stos liczb całkowitych. Klasa posiada także metodę int calculate(String onp) pozwalającą na obliczenie wyrażenia ONP reprezentowanego przez łańcuch znaków onp. W razie potrzeby zdefiniuj pomocnicze metody prywatne.

Załącznik nr 1

Algorytm 1 (zamiana wyrażenia algebraicznego na ONP)

POWTARZAJ wczytaj **słowo** z wejścia

1) Jeżeli przeczytane słowo to **stała/zmienna** to przesyłana jest na wyjście.

2) Jeżeli przeczytane słowo to **operator**:

"**(**": Nawias otwierający, dopisywany jest zawsze na stos.

"**)**": Nawias zamykający, należy odczytać ze stosu i wyświetlić na wyjściu wszystkie operatory do znaku "(", a następnie usunąć "(" ze stosu

"*","+","-","/","%": JEŻELI "priorytet operatora wejściowego jest wyższy od priorytetu operatora na stosie" lub "stos jest pusty" to należy dodać do stosu operator. W PRZECIWNYM RAZIE należy odczytać ze stosu i przesłać na wyjście kolejne operatory o priorytecie wyższym lub równym wejściowemu operatorowi, po czym dodać operator wejściowy na stos.

3) Jeżeli czytane dane z wejścia skończyły się to należy odczytać i przesłać całą zawartość stosu na wyjście. Następnie zakończyć działanie algorytmu.

Załącznik nr 2

Algorytm 2 (obliczenie wyrażenia ONP)

Warunki początkowe:

- Wejście z którego czytane jest wyrażenie ONP
- Pusty stos typu całkowitoliczbowego

POWTARZAJ wczytaj **słowo** z wejścia:

1) Jeżeli na wejściu jest liczba to przekazywana jest na stos.
2) Jeżeli na wejściu pojawia się znak operacji **@** to:
   2.1) pobierane są argumenty ze stosu: niech **a2** oznacza pierwszy pobrany argument a **a1** drugi pobrany,
   2.2)wykonywane jest działanie (ważna kolejność argumentów) **w1=a1@a2**, po czym wynik **w1** jest umieszczany na stosie.

Po zakończeniu czytania danych, wynik ONP znajduje się na stosie.

Załącznik nr 3

Znak	Priorytet
(	pusty (0)
+, -,)	1
***, %, /**	2