# Lekcja – stos

Stos (ang. stack) i kolejka (ang. queue) to struktury danych, które pozwalają na przechowywanie i zarządzanie elementami w szczególny sposób. Ich działanie można opisać przy użyciu analogii do życia codziennego: stos przypomina stos talerzy, gdzie możemy dodawać i usuwać talerze tylko z góry, natomiast kolejka przypomina kolejkę w sklepie, gdzie pierwsza osoba, która dołączy do kolejki, jest pierwszą obsłużoną.

## Stos

Stos działa na zasadzie LIFO (Last In, First Out), co oznacza, że ostatni element dodany do stosu jest pierwszym elementem, który zostanie z niego usunięty. W Javie stos można zaimplementować za pomocą klasy Stack<E>.

Przykład:

**Stack<Integer> stos = new Stack<>();**

**// Dodanie elementów do stosu**

**stos.push(1);**

**stos.push(2);**

**stos.push(3);**

**// Teraz stos wygląda tak: [1, 2, 3], gdzie 3 jest na górze stosu**

**// Usunięcie elementu ze stosu (usuwamy element z góry stosu)**

**int usunietyElement = stos.pop(); // usunietyElement = 3**

**// Teraz stos wygląda tak: [1, 2]**

## Zastosowanie

**1. Zarządzanie wywołaniami funkcji i rekurencja**

Stos wywołań (call stack) jest kluczowym elementem w językach programowania, służącym do śledzenia punktów, w których program powinien powrócić po zakończeniu wykonywania funkcji. Kiedy funkcja wywołuje inną funkcję, adres powrotu i parametry są zapisywane na stosie. Po zakończeniu wykonywania funkcji, te informacje są wykorzystywane do powrotu do miejsca wywołania.

**2. Nawigacja w przeglądarce (Cofnij/Do przodu)**

Stosy są wykorzystywane do implementacji funkcji nawigacyjnych w przeglądarkach internetowych, takich jak przyciski "Cofnij" i "Do przodu". Każda odwiedzona strona jest umieszczana na stosie. Kiedy użytkownik klika "Cofnij", przeglądarka pobiera adres URL z góry stosu i ładuje poprzednią stronę.

**3. Analiza składniowa w kompilatorach**

Kompilatory używają stosów do analizy składniowej kodu źródłowego programów. Podczas przetwarzania wyrażeń i instrukcji, stos pomaga w śledzeniu operatorów i operandów, umożliwiając właściwe ich zagnieżdżenie i ewaluację.

**4. Odwracanie ciągów znaków**

Stos może być użyty do odwrócenia ciągu znaków. Każdy znak ciągu jest umieszczany na stosie, a następnie są one pobierane jeden po drugim, co skutkuje odwróceniem kolejności znaków.

**5. Sprawdzanie poprawności nawiasów**

Algorytm używający stosu może służyć do sprawdzania, czy nawiasy w wyrażeniu matematycznym lub kodzie programu są poprawnie sparowane i zagnieżdżone. Każdy otwierający nawias jest umieszczany na stosie, a przy każdym zamykającym nawiasie sprawdza się, czy na górze stosu znajduje się odpowiadający mu otwierający nawias.

**6. Przechodzenie drzew (np. w głąb)**

Algorytmy przechodzenia drzew, takie jak przechodzenie w głąb (DFS), często wykorzystują stos do śledzenia wierzchołków, które mają zostać odwiedzone.

**7. Operacje na liczbach w notacji polskiej (odwrotnej)**

Stosy są używane do obliczeń wyrażeń matematycznych zapisanych w notacji polskiej odwrotnej (RPN), gdzie operator występuje po operandach. Dzięki stosowaniu stosu, operacje matematyczne mogą być wykonywane w odpowiedniej kolejności bez potrzeby stosowania nawiasów.

## Zadanie 1

Jakiej innej struktury danych możemy użyć w celu zaimplementowaniu stosu – odpowiedź uzasadnij?

## Zadanie 2 - sprawdzanie poprawności użycia nawiasów

Napisz program, który sprawdza, czy w danym wyrażeniu nawiasy (okrągłe (), kwadratowe [] i klamrowe {}) są poprawnie sparowane i zagnieżdżone. Program powinien używać stosu do śledzenia otwartych nawiasów i zapewnić, że każdy z nich jest prawidłowo zamknięty w odpowiedniej kolejności.

Ala ([ma]) {kota} i psa i cos jeszcze – przykład poprawnego zagnieżdżenia

Ala ([ma]) {kota]} i psa i cos jeszcze – przykład niepoprawnego zagnieżdżenia

## Zadanie 3 – odwracanie ciągu znaków

Użyj stosu do odwrócenia ciągu znaków. Program powinien odczytać ciąg znaków, umieścić każdy znak na stosie, a następnie usunąć znaki ze stosu, efektywnie odwracając kolejność ciągu.

„a”, „b”, „c” -> „c”, „b”, „a”

## Zadanie 4 – Obliczanie wyrażeń w Odwrotnej Notacji Polskiej (ONP)

Napisz program, który oblicza wartość wyrażenia podanego w Odwrotnej Notacji Polskiej (ONP). Program powinien odczytywać wyrażenie składające się z liczb i operatorów, stosując stos do przechowywania liczb i wykonania na nich operacji matematycznych w odpowiedniej kolejności.

3 4 + 2 \* 7 /

Zacznij od lewej strony, pierwsze dwa liczby to 3 i 4, a następnie operator +. Dodajemy 3 i 4, co daje 7.

7 2 \* 7 /

Następnie mamy 2 i operator \*. Mnożymy 7 (wynik poprzedniej operacji) przez 2, co daje 14.

14 7 /

Ostatecznie, mamy 7 i operator /. Dzielimy 14 przez 7, co daje wynik 2.

Więc wartość wyrażenia 3 4 + 2 \* 7 / w Odwrotnej Notacji Polskiej to 2.