# LIFO databeheer

# stack

### Voorbeeld

### Te gebruiken bij:

-LIFO data beheer

### Eigenschappen:

-

### Info

#### stack

Ja, je kunt "stacken" in programmeren vergelijken met het concept van een stapel (stack) in de computerwetenschap.

In de context van programmeren is een "stack" een datastructuur die elementen op een specifieke manier beheert. Het volgt het LIFO (Last In, First Out) principe, wat betekent dat het laatst toegevoegde element als eerste wordt verwijderd. Dit komt overeen met het idee van een stapel objecten waar je objecten bovenop kunt plaatsen en alleen het bovenste object kunt verwijderen.

Bij het "stacken" van acties in programmeren, voeg je acties toe aan een stapel (stack) en voer je ze vervolgens uit in omgekeerde volgorde. Dit kan handig zijn in situaties waarin je een reeks acties wilt uitvoeren en later wilt terugkeren naar een eerdere toestand door de acties in omgekeerde volgorde ongedaan te maken.

Hier is een eenvoudig voorbeeld in pseudocode om het concept van stacken te illustreren:

arduino

stack = lege\_stack() // Maak een lege stack

stack.push("Actie 1") // Voeg Actie 1 toe aan de stack

stack.push("Actie 2") // Voeg Actie 2 toe aan de stack

stack.push("Actie 3") // Voeg Actie 3 toe aan de stack

while stack.is\_not\_empty():

actie = stack.pop() // Haal de bovenste actie van de stack

voer\_actie\_uit(actie) // Voer de actie uit

// De acties worden in omgekeerde volgorde uitgevoerd: Actie 3, Actie 2, Actie 1

Dus, in programmeertaaltermen, kun je "stacken" vergelijken met het werken met een stapel (stack) waarop je elementen plaatst en verwijdert volgens het LIFO-principe.

1. Functieoproepen en -retouren: Wanneer een functie wordt opgeroepen, wordt de huidige uitvoeringscontext op de stack geplaatst. Wanneer de functie klaar is met uitvoeren, wordt de context van de stack verwijderd en wordt de uitvoering hervat vanaf het punt waar de functie werd opgeroepen. Dit wordt mogelijk gemaakt door de call stack.
2. Undo/Redo-functionaliteit: Stacks worden vaak gebruikt bij het implementeren van undo/redo-functionaliteit in toepassingen. Elke actie die kan worden teruggedraaid, wordt op de stack geplaatst. Als een undo-operatie wordt uitgevoerd, wordt de laatst toegevoegde actie van de stack verwijderd en ongedaan gemaakt. Voor redo wordt een verwijderde actie opnieuw toegevoegd aan de stack.
3. Backtracking: Bij bepaalde algoritmen, zoals diepte-eerst zoeken in grafen, kunnen stacks worden gebruikt om de huidige staat of paden bij te houden. Hierdoor kan men terugkeren naar vorige knooppunten of paden en alternatieve routes verkennen.
4. Parserimplementatie: Stacks worden vaak gebruikt bij het implementeren van parsers, zoals voor het analyseren van wiskundige uitdrukkingen of het valideren van syntaxis. Een stack kan worden gebruikt om tijdelijke gegevens of operatoren bij te houden tijdens het parsen van de invoer.