Zásobník (Stack)

Zásobník (anglicky *Stack*) je abstraktní datová struktura, která slouží k ukládání prvků v lineárním uspořádání. Je fundamentální součástí informatiky a funguje na principu **LIFO** (Last In, First Out), což v překladu znamená „poslední dovnitř, první ven“.

Tuto strukturu si lze představit jako **tos talířů v jídelně**:

- Když přinesete nový talíř, položíte ho **nahoru** na ostatní.
- Když si chcete vzít talíř, musíte vzít ten, který je **nahoře** (ten, který byl položen jako poslední).
- Není možné bezpečně vytáhnout talíř ze spodní části, aniž byste odebrali ty nad ním.

Klíčové vlastnosti

- **Lineární struktura:** Prvky jsou řazeny za sebou.
- **Omezený přístup:** Přistupovat lze pouze k prvku na vrcholu (Top).
- **Efektivita:** Vložení a odebrání prvku má konstantní časovou složitost **O(1)**.

Hlavní operace

Každá implementace zásobníku musí podporovat následující základní operace:

Operace	Název	Popis
Vložení	``push()``	Přidá nový prvek na **vrchol** zásobníku.
Odebrání	``pop()``	Odebere prvek z **vrcholu** zásobníku a vrátí jeho hodnotu.
Náhled	``peek()`` nebo ``top()``	Vrátí hodnotu prvku na vrcholu, ale **neodebere** jej.
Je prázdný?	``isEmpty()``	Vrátí *true*, pokud v zásobníku nejsou žádné prvky.
Velikost	``size()``	Vrátí aktuální počet prvků (volitelné).

Implementace zásobníku

V programování lze zásobník implementovat dvěma hlavními způsoby:

1. Pomocí pole (Array)

Jedná se o statickou implementaci.

- **Výhody:** Rychlý přístup, paměťová efektivita (není třeba ukládat ukazatele).
- **Nevýhody:** Pevná velikost. Pokud se zásobník naplní, dojde k chybě *Stack Overflow* (pokud není pole dynamicky realokováno).

2. Pomocí spojového seznamu (Linked List)

Jedná se o dynamickou implementaci.

- **Výhody:** Velikost je omezena pouze pamětí počítače, roste podle potřeby.
- **Nevýhody:** Vyšší paměťová náročnost (každý prvek potřebuje extra paměť pro odkaz na další prvek).

Příklad implementace (Pseudokód)

```
class Stack {
    pole data[100];
    int top = -1; // Ukazatel na vrchol (prázdný zásobník)

    void push(x) {
        top++;
        data[top] = x;
    }

    int pop() {
        if (top == -1) return error;
        hodnota = data[top];
        top--;
        return hodnota;
    }
}
```

Využití zásobníku v praxi

Zásobník je jednou z nejpoužívanějších struktur, často i tam, kde to uživatel nevidí.

1. Zásobník volání (Call Stack)

Pravděpodobně nejdůležitější využití. Když program zavolá funkci:

1. 1. Adresa návratu a lokální proměnné jsou uloženy na zásobník (tzv. **Stack Frame**).
2. 2. Pokud tato funkce zavolá další funkci, přidá se na vrchol nový Stack Frame.
3. 3. Jakmile funkce skončí, její rámec je odstraněn (Pop) a řízení se vrací funkci pod ní.

> **Poznámka:** Pokud dojde k nekonečné rekurzi, paměť zásobníku se vyčerpá a program spadne na chybu **Stack Overflow**.

2. Krok zpět (Undo/Redo)

Textové editory (Word, VS Code) používají dva zásobníky pro sledování změn:

- Když napíšete text, akce se uloží do zásobníku **Undo**.
- Když stisknete *Ctrl+Z*, akce se přesune ze zásobníku **Undo** do zásobníku **Redo**.

3. Kontrola závorek (Syntax Parsing)

Kompilátory používají zásobník pro kontrolu, zda jsou závorky ve zdrojovém kódu správně spárovány.

- Otevírací závorka ``(``, ``{``, ``[`` → **Push**.
- Zavírací závorka ``)``, ``}``, ``]`` → **Pop** (a zkontroluje se shoda).
- Pokud je na konci zásobník prázdný, syntaxe je správná.

4. Vyhodnocování výrazů (RPN)

Zásobník se používá pro vyhodnocování matematických výrazů zapsaných v **Reverzní polské notaci** (např. ``3 4 +`` místo ``3 + 4``).

Srovnání s Frontou (Queue)

Je důležité nezaměňovat zásobník s frontou.

Vlastnost	Zásobník (Stack)	Fronta (Queue)
Princip	**LIFO** (Last In, First Out)	**FIFO** (First In, First Out)
Přidávání	Push (na vrchol)	Enqueue (na konec)
Odebírání	Pop (z vrcholu)	Dequeue (ze začátku)
Analogie	Tos talířů	Fronta u pokladny

Časová složitost (Big O)

Operace nad zásobníkem jsou extrémně rychlé, protože se vždy pracuje pouze s jedním koncem datové struktury.

- **Přístup (Access):** *O(n)* (Zásobník není určen k prohledávání, pro přístup k n-tému prvku musíme odebrat vše nad ním).
- **Hledání (Search):** *O(n)*
- **Vložení (Push):** *O(1)*
- **Smazání (Pop):** *O(1)*

Autorem tohoto textu je AI asistent (2025).