

Typografie a publikování – 5. projekt

Lineárny zoznam

Natália Bubáková

VUT FIT

9. mája 2021

Prehľad

- 1 Dátové štruktúry
- 2 Úvod do lineárneho zoznamu
- 3 Typy lineárnych zoznamov
- 4 Štruktúra
- 5 Praktické využitie
- 6 Zhrnutie

Dátové štruktúry

Dátová štruktúra je ...

- logická reprezentácia dát, poradie jej prvkov a vzťahy medzi nimi
- spôsob uloženia jej prvkov a toho ako s nimi možno pracovať
- môže byť *lineárna* alebo *nelineárna*

Typy dátových štruktúr:

- pole (*array*)
- front (*queue*)
- slovník (*dictionary*)
- zásobník (*stack*)
- halda (*heap*)
- strom (*tree*)
- bunka s využitím smerníka
- dátum a čas
- množina (*set*)
- trieda (*class*)
- vektor (*vector*)
- graf
- zobrazenie (*map*)
- zoznam (*list*)
- **spájaný zoznam** (*linked list*)

Prečo práve *lineárny zoznam* ?

Medzi najčastejšie využívané dátové štruktúry patrí *pole*

Pole (array)

- + je ľahké na pochopenie
- + jednoduchý prístup k jednotlivým prvkom

ale ...

- ukladá sa len do súvislej pamäti (*contiguous memory*)
- má pevne určenú dimenziu
- raz po uvedení jeho veľkosti je táto nemenná
- vloženie a zmazanie jeho prvkov je veľmi pracné

... a práve tieto obmedzenia pokrýva lineárny zoznam (*linked list*)

A čo to je ten *lineárny zoznam* !?

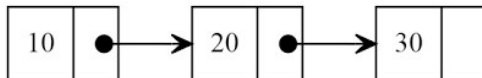
Lineárny / spájaný zoznam (*linked list*)

- je lineárna štruktúra, ktorá pozostáva zo sekvencie prvkov, zvaných *uzle*
- každý prvok pozostáva z dvoch častí
 - 1 obsah uložených dát v danom prvku
 - 2 odkaz (adresa) na nasledujúci prvok



Príklad:

Jednodúchý lineárny zoznam čísiel 10, 20, 30 sa dá zobrazit' ako:



Typy lineárných zoznamov

Jednosmerný lineárny zoznam (*singly linked list*)


- je sekvencia uzlov pozostávajúcich z dát a jedného ukazovateľa na nasledujúci uzol, tzv. *next* $\square \rightarrow \square \rightarrow \square \rightarrow \dots$

Obojsmerný lineárny zoznam (*doubly linked list*)

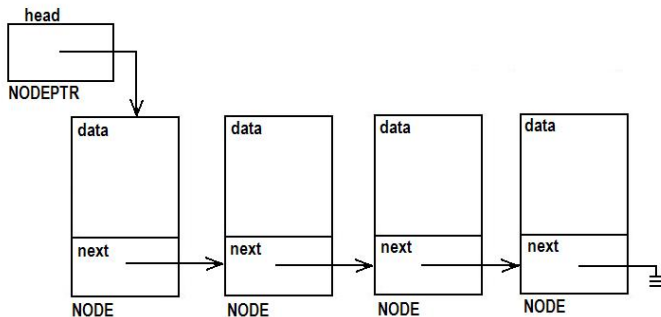
- je sekvencia uzlov pozostávajúcich z dát a dvoch ukazovateľov na predošlý a nasledujúci uzol, tzv. *next* a *previous* $\square \rightleftarrows \square \rightleftarrows \square \rightleftarrows \dots$

Cyklický lineárny zoznam (*circular linked list*)

- je sekvencia uzlov, kde posledný uzol drží ukazovateľ na prvý uzol, a tak tvorí uzavretý „logický kruh“ $\rightarrow \square \rightarrow \square \rightarrow \dots \rightarrow \square \rightarrow$

Jednosmerný lineárny zoznam predstavuje základ pre všetky spomenuté, preto sa pre účel tejto prezentácie ďalej venujeme práve tomuto typu: 

Štruktúra



Sekvencia prvkov

pozostáva nie len z uzlov so samotnými dátami, ale tiež

- z referenčného ukazovateľa na začiatok zoznamu, tzv. *head*
- z posledného uzla, tzv. *tail* ukazujúc na koniec, teda *NULL*

Náhľad na kód

Zostavenie zoznamu (z 3 uzlov):

```
int main(){  
    struct Node* head    = NULL;  
    struct Node* second = NULL;  
    struct Node* third  = NULL;
```

```
    head    = (struct Node*)malloc(sizeof(struct Node));  
    second  = (struct Node*)malloc(sizeof(struct Node));  
    third   = (struct Node*)malloc(sizeof(struct Node));
```

```
    head    -> data = 1;  
    second  -> data = 2;  
    third   -> data = 3;  
    return 0;  
}
```

```
    head    -> next = second;  
    second  -> next = third;  
    third   -> next = NULL;
```

Štruktúra uzla:

```
struct Node{  
    int data;  
    struct Node *next;  
};
```


Praktické využitie

Vloženie nového uzlu (*insertion*)

```
void push(struct Node** head_ref, int new_data){
```

- 1 alokovať priestor pre nový uzol

```
    struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));
```

- 2 vložiť dáta do nového uzlu

```
    new_node->data = new_data;
```

- 3 ukazateľom ho pripojiť na zoznam

```
    new_node->next = (*head_ref);
```

- 4 uistiť sa, že *head* ukazuje na začiatok

```
    (*head_ref) = new_node;
```

```
}
```

Vymazanie existujúceho uzlu (*deletion*)

```
void deleteNode(struct Node** head_ref, int key){
```

- 1 vytvoriť si pomocnú premennú zo začiatku zoznamu

```
struct Node *temp = *head_ref, *prev;
```
- 2 skontrolovať či nie je hľadaný kľúč v prvom prvku

```
if (temp != NULL && temp->data == key)
{*head_ref = temp->next; free(temp);return;}
```
- 3 nájsť daný prvok podľa kľúču

```
while (temp != NULL && temp->data != key)
{prev = temp;temp = temp->next;}
```
- 4 overiť si či sa prvok nachádza v zozname

```
if (temp == NULL) return;
```
- 5 presunúť ukazovateľ tak, aby preskočil nežiadany prvok

```
prev->next = temp->next;
```
- 6 uvoľniť alokovanú pamäť

```
free(temp); }
```

Zhrnutie





Výhody

- + Lineárny zoznam je v pamäti dynamicky alokovaný, teda netreba vopred vedieť jeho veľkosť a možno ho kedykoľvek pohodlne rožšíriť.
- + Vloženie a vymazanie uzlu sa da jednoducho zrealizovať.
- + Vhodné na implementáciu fronty, grafu či zásobníku.
- + Jeho štruktúra výrazne znižuje prístupový čas.

Nevýhody

- Ukazovatele zaberajú navyše priestor v pamäti.
- K prvkom sa nedá pristupovať náhodne, treba senkvenčne prejsť každým jedným uzlom.

Použité zdroje

-  Wikipédia: Údajová štruktúra
<https://sk.wikipedia.org/wiki>
-  Study tonight: Introduction to linked list
<https://www.studytonight.com>
-  Slide Share: Linked list in data structure
<https://www.slideshare.net>
-  Geeks for geeks: Linked List Data Structure
<https://www.geeksforgeeks.org>