

Dátové štruktúry a algoritmy

Gabriel Juhás

Dátové struktúry

1. Jednorozmerné statické pole – one dimensional static array
2. Viacrozmerné statické pole – multidimensional static array (2D ..., matica - matrix)
3. Jednorozmerné dynamické pole – one dimensional dynamic array
4. Viacrozmerné dynamické pole – multidimensional dynamic array (2D ...)
5. Retázec - string

Dátové štruktúry

- 6. Ohraničený zásobník – bounded stack
- 7. Neohraničený zásobník – unbounded stack
- 8. Ohraničená fronta – bounded queue
- 9. Neohraničená fronta – unbounded queue

Dátové štruktúry

- 10. Jednosmerný zretázený zoznam – single linked list
- 11. Obojsmerný zretázený zoznam – doubly linked list
- 12. Jednosmerný cyklický zretázený zoznam – circular single linked list
- 13. Obojsmerný cyklický zretázený zoznam – circular doubly linked list

Dátové struktúry

14. Množina – set (a hash set)

15. Mapa – map (a hash table)

16. Graf – graph

17. Tree – strom



Dátové štruktúry

18. Binárny strom – binary tree

19. Binárny vyhľadávací strom – binary search tree

20. Balancovaný binárny strom – balanced binary tree

21. Balancovaný binárny vyhľadávací strom – balanced binary search tree

22. Prefixový strom - prefix tree – trie

23. Halda a prioritná fronta – heap and priority queue

(Typické) Algoritmy, princípy a operácie (na dátových štruktúrach)

- Jednorozmerné statické pole – one dimensional static array – **prechádzanie, vyhľadávanie prvku, triedenie (zoraďovanie prvov), porovnávanie polí ...**
- Viacrozmerné statické pole – multidimensional static array (2D ..., matica - matrix) – **prechádzanie, transponovanie**
- Jednorozmerné dynamické pole – one dimensional dynamic array - **prechádzanie, vyhľadávanie prvku, triedenie (zoraďovanie prvov), porovnávanie polí ...**
- Viacrozmerné dynamické pole – multidimensional dynamic array (2D ...) - **prechádzanie, transponovanie**
- Retázec – string – **prefix, postfix, substring, konkaténácia**

(Typické) Algoritmy, princípy a operácie (na dátových štruktúrach)

- Ohraničený zásobník – bounded stack **LIFO (last in first out)** – **push, pop, peek, isempty, isfull, stack overflow**
- Neohraničený zásobník – unbounded stack **LIFO (last in first out)** – **push, pop, peek, isempty**
- Ohraničená fronta – bounded queue **FIFO, enqueue, dequeue, front, isempty, isfull**
- Neohraničená fronta – unbounded queue **FIFO, enqueue, dequeue, front, isempty**

(Typické) Algoritmy, princípy a operácie (na dátových štruktúrach)

- Jednosmerný zretázený zoznam – **prechádzanie, vyhľadávanie prvku, vkladanie prvku na začiatok, na koniec, podľa abecedy**
- Obojsmerný zretázený zoznam – **obojsmerné prechádzanie (traversal)**
- Jednosmerný cyklický zretázený zoznam – circular single linked list **cyklické prechádzanie, vkladanie**
- Obojsmerný cyklický zretázený zoznam – circular doubly linked list **obojsmerné cyklické prechádzanie vkladanie**

(Typické) Algoritmy, princípy a operácie (na dátových štruktúrach)

- Množina – set (a hash set) – **zjednotenie, prienik, vyhľadanie prvku, prechádzanie, vkladanie, mazanie prvku, hashovacie funkcie**
- Mapa – map (a hash table) – **prechádzanie, vkladanie, mazanie, vyhľadávanie**
- Graf – graph – **prechádzanie, vkladanie vrcholu, vkladanie hrany, mazanie vrcholu a mazanie hrany, prehľadávanie (do hĺbky a do šírky), váhované grafy, najkratšia cesta, zafarbitelnosť, planárne grafy, kostra – spanning tree, linearizácia – topological sorting orientovaných grafov**
- Tree – strom – **prechádzanie, vkladanie, mazanie, zisťovanie hĺbky**

(Typické) Algoritmy, princípy a operácie (na dátových štruktúrach)

- Binárny strom – **binary tree** (pre order, in order, post order, level order, reverse level order) **prechádzanie, hľadanie...**
- Binárny vyhľadávací strom – binary search tree **hľadanie prechádzanie vkladanie mazanie**
- Balancovaný binárny strom – balanced binary tree
- Balancovaný binárny vyhľadávací strom – balanced binary search tree **balancovanie pri vkladaní a mazaní**
- Prefixový strom - prefix tree – trie - **vkladanie, hľadanie prefixu**
- Hald a prioritná fronta – heap and priority queue - **vkladanie, hľadanie mazanie**

Dátové struktúry

- Jednorozmerné statické pole – one dimensional static array

```
4 | int st_pole[6];  
5 |  
6 | = for (int i = 0; i < 6; i++) {  
7 |     printf("Zadaj st_pole[%d]: ", i);  
8 |     scanf("%d", st_pole + i); // &st_pole[i]  
9 | }
```

index	0	1	2	3	4	5
hodnota	12	17	21	47	2	18

Dátové štruktúry

- Viacrozmerné statické pole – multidimensional static array (2D ..., matica - matrix)

```
5      int st_pole[3][4];
6
7      for (int i = 0; i < 3; i++)
8      {
9          for (int j = 0; j < 4; j++)
10         {
11             printf("zadaj st_pole[%d][%d] ", i,j);
12             scanf("%d", st_pole[i] + j);
13         }
14     }
```

st_pole

1	2	3	3	4	5	6	3	7	8	9	0
---	---	---	---	---	---	---	---	---	---	---	---

0	0	1	2	3
	1	2	3	3
1	0	1	2	3
	4	5	6	3
2	0	1	2	3
	7	8	9	0

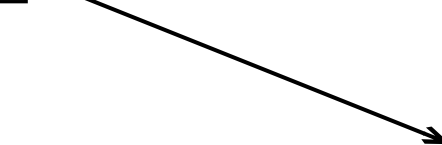
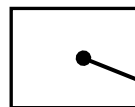
Dátové struktúry

- Jednorozmerné dynamické pole – one dimensional dynamic array

```
6      int *dyn_pole, pocet;  
7      printf("zadaj pocet prvkov pola ");  
8      scanf("%d", &pocet);  
9      printf("velkost int je %d\n", sizeof(int));  
10     if ((dyn_pole = (int*) malloc(pocet * sizeof(int))) != NULL)  
11     {   for (int i = 0; i < pocet; i++)  
12         {   printf("zadaj dyn_pole[%d] ", i);  
13             scanf("%d", dyn_pole + i);  
14         }  
15     }
```

```
5      int *dyn_pole, pocet;  
6      printf("Zadaj pocet prvkov \n");  
7      scanf("%d", &pocet);  
8      dyn_pole = new int[pocet];  
9      for (int i = 0; i < pocet; i++) {  
10         printf("Zadaj dyn_pole[%d]: ", i);  
11         scanf("%d", dyn_pole + i); // &dyn_pole[i]  
12     }
```

dyn_pole



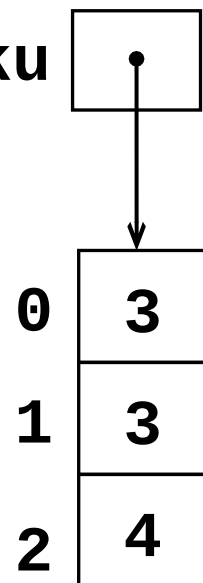
0	1	2	3
7	8	9	0

Dátové štruktúry

- Viacrozmerné dynamické pole
multidimensional dynamic array

3 pocetriadkov

pocet_vriadku

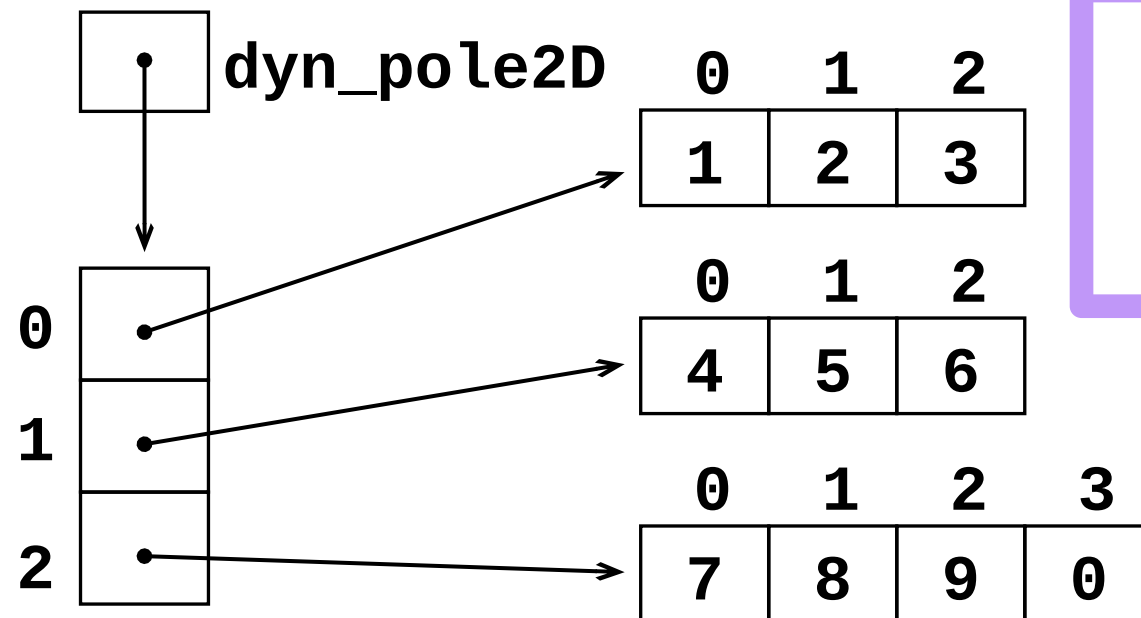


```
int **dyn_pole2D, *pocet_vriadku, pocetriadkov, i, j;

printf("Zadaj pocet riadkov ");
scanf("%d", &pocetriadkov);

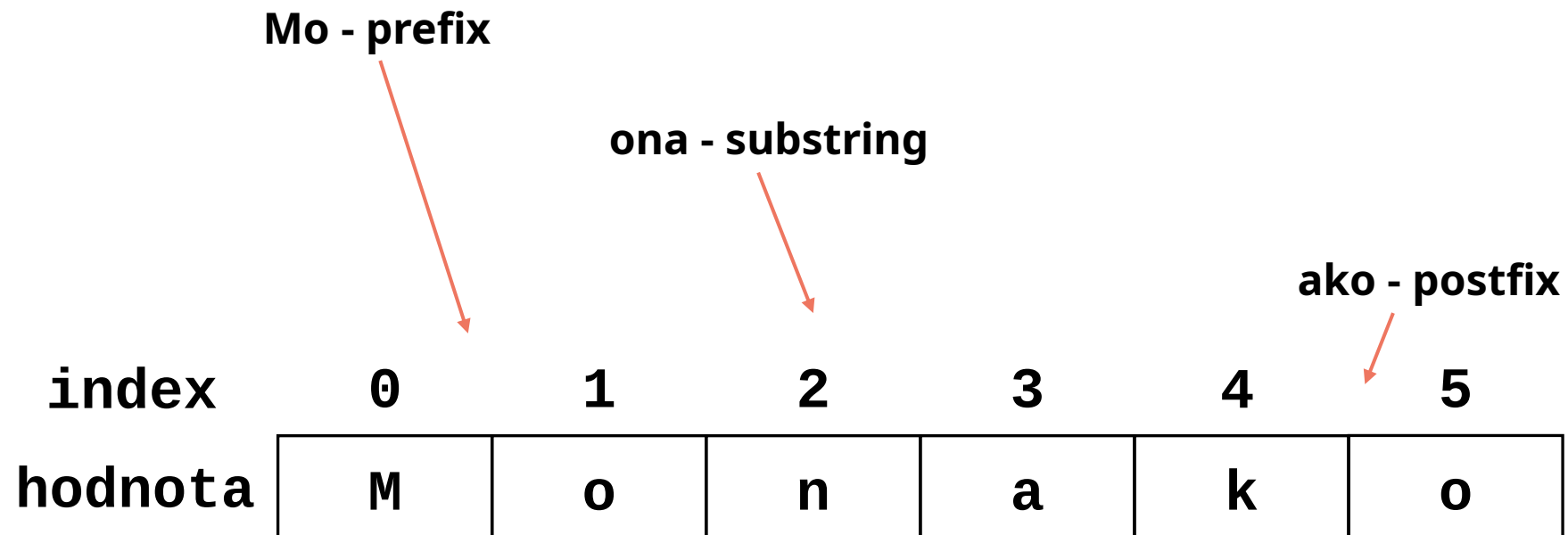
dyn_pole2D = (int**) malloc(pocetriadkov * sizeof(int*));
pocet_vriadku = (int*) malloc(pocetriadkov * sizeof(int));

for (i = 0; i < pocetriadkov; i++) {
    printf("zadaj pocet prvkov v riadku %d ", i);
    scanf("%d", pocet_vriadku + i);
    dyn_pole2D[i] = (int*) malloc(pocet_vriadku[i] * sizeof(int));
    for (j = 0; j < pocet_vriadku[i]; j++) {
        printf("Zadaj prvok pola dyn_pole2D[%d][%d] = ", i, j);
        scanf("%d", &(dyn_pole2D[i][j]));
    }
}
```



Dátové štruktúry

- String – reťazec



The diagram shows a string 'monako' stored in an array. The array has six cells, indexed from 0 to 5. The characters are 'M', 'o', 'n', 'a', 'k', and 'o' respectively. Three annotations with red arrows point to specific parts of the string: 'Mo - prefix' points to the first two characters ('M', 'o'), 'ona - substring' points to the last three characters ('n', 'a', 'k'), and 'ako - postfix' points to the last three characters ('a', 'k', 'o').

index	0	1	2	3	4	5
hodnota	M	o	n	a	k	o

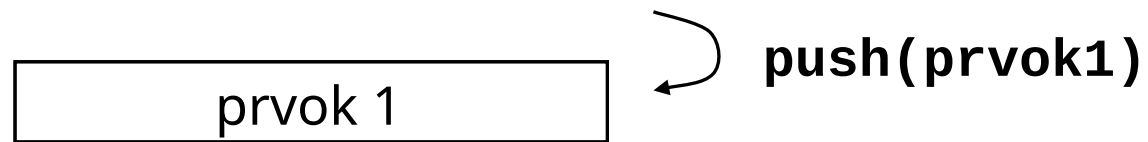
Dátové struktúry

- Ohraničený zásobník – bounded stack
- Neohraničený zásobník – unbounded stack

↪ `push(prvok1)`

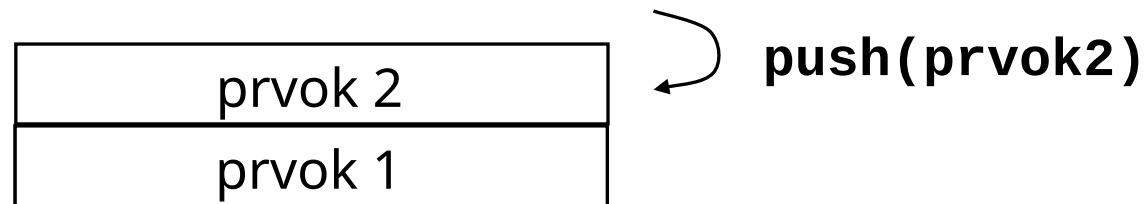
Dátové struktúry

- Ohraničený zásobník – bounded stack
- Neohraničený zásobník – unbounded stack



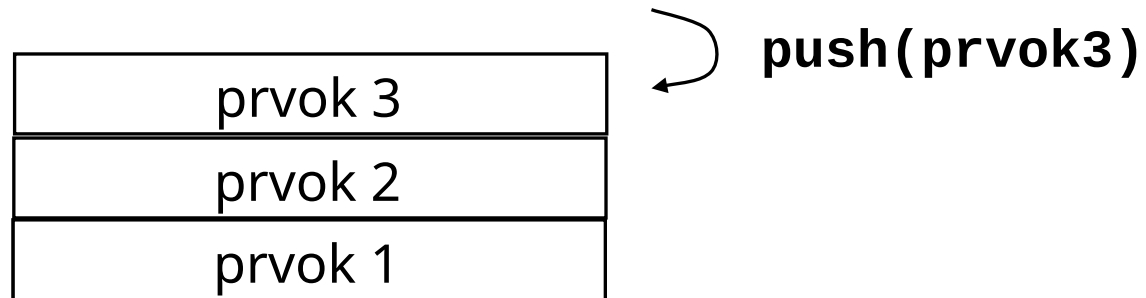
Dátové struktúry

- Ohraničený zásobník – bounded stack
- Neohraničený zásobník – unbounded stack



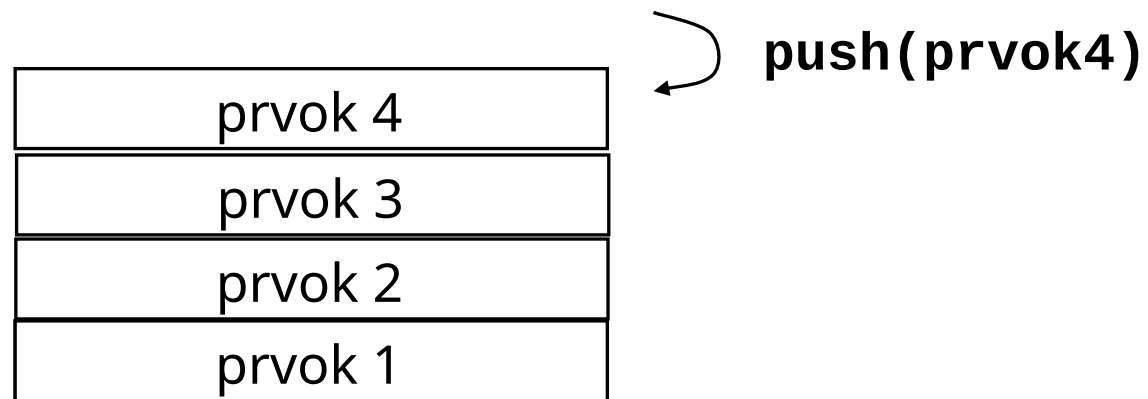
Dátové struktúry

- Ohraničený zásobník – bounded stack
- Neohraničený zásobník – unbounded stack



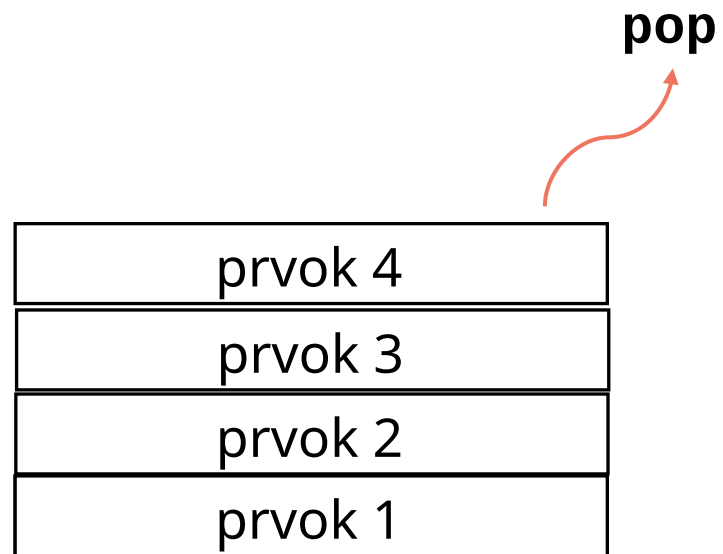
Dátové struktúry

- Ohraničený zásobník – bounded stack
- Neohraničený zásobník – unbounded stack




Dátové struktúry

- Ohraničený zásobník – bounded stack
- Neohraničený zásobník – unbounded stack

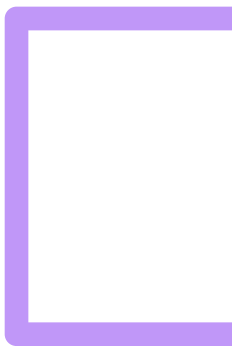


Dátové štruktúry

- Ohraničený zásobník – bounded stack
- Neohraničený zásobník – unbounded stack

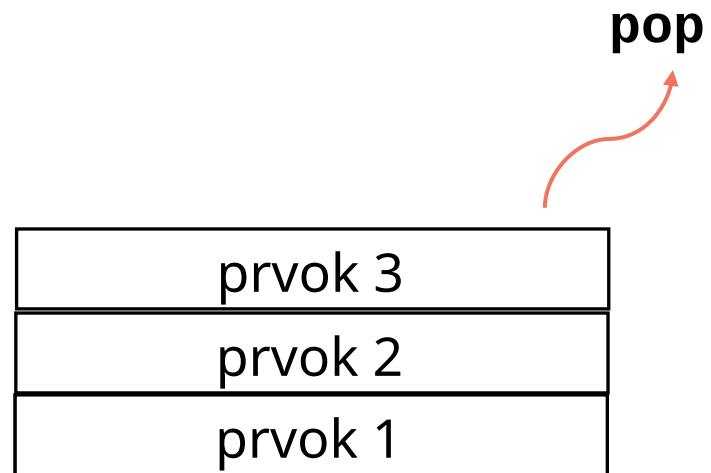


prvok 3
prvok 2
prvok 1



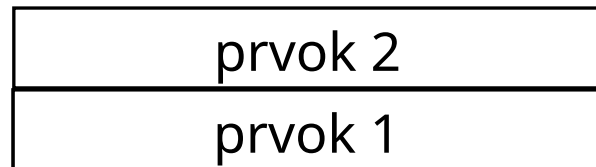
Dátové štruktúry

- Ohraničený zásobník – bounded stack
- Neohraničený zásobník – unbounded stack



Dátové struktúry

- Ohraničený zásobník – bounded stack
- Neohraničený zásobník – unbounded stack



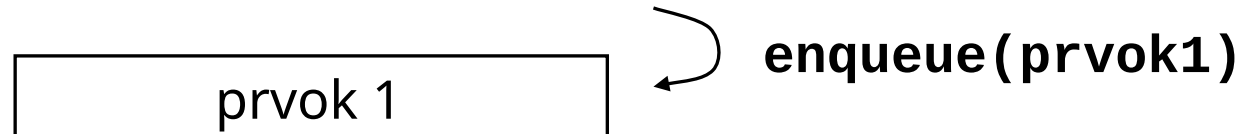
Dátové štruktúry

- Ohraničená fronta – bounded queue
- Neohraničená fronta – unbounded queue

↪ `enqueue(prvok1)`

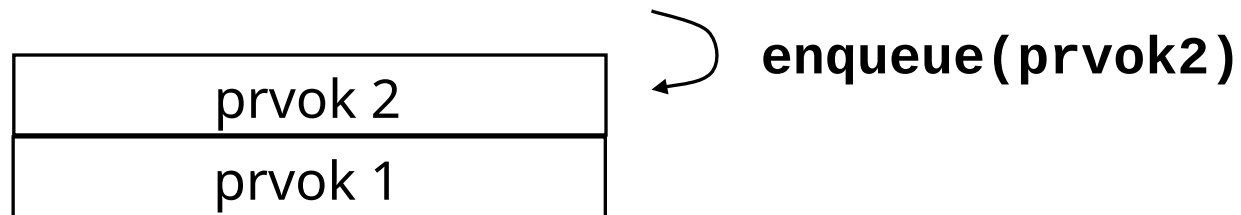
Dátové štruktúry

- Ohraničená fronta – bounded queue
- Neohraničená fronta – unbounded queue



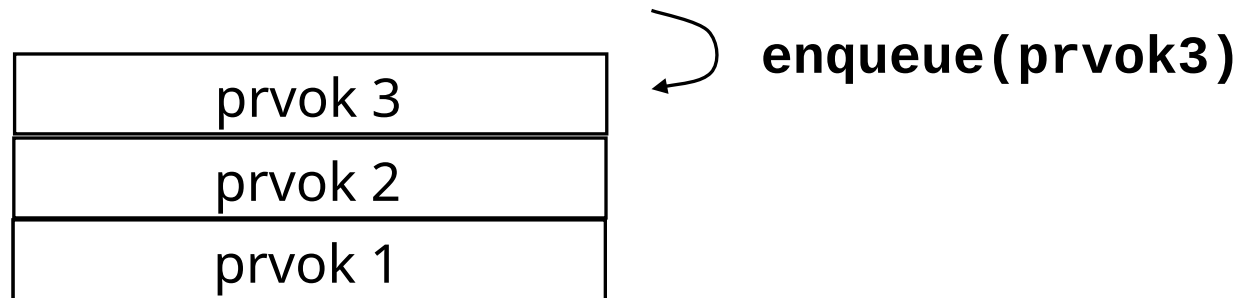
Dátové štruktúry

- Ohraničená fronta – bounded queue
- Neohraničená fronta – unbounded queue



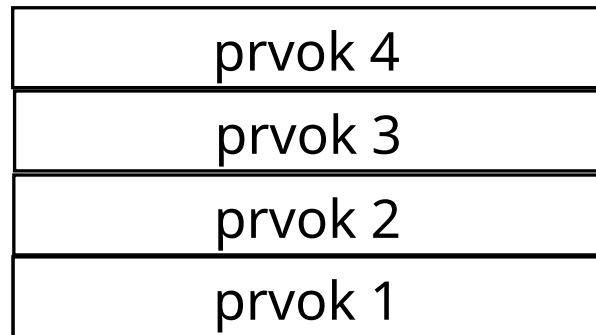
Dátové štruktúry

- Ohraničená fronta – bounded queue
- Neohraničená fronta – unbounded queue



Dátové struktúry

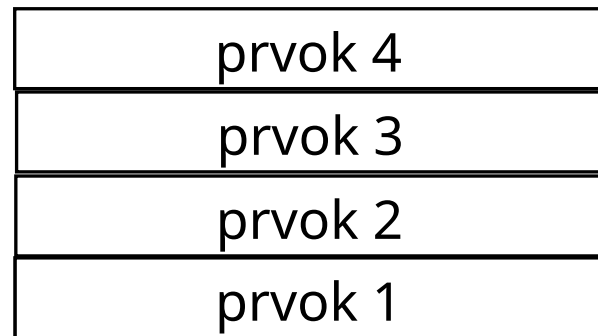
- Ohraničená fronta – bounded queue
- Neohraničená fronta – unbounded queue



`enqueue(prvok4)`

Dátové štruktúry

- Ohraničená fronta – **bounded queue**
- Neohraničená fronta – **unbounded queue**



 **dequeue**

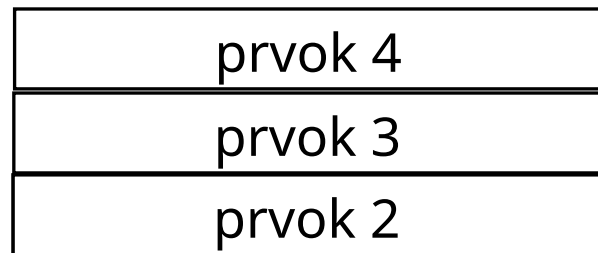
Dátové štruktúry

- Ohraničená fronta – bounded queue
- Neohraničená fronta – unbounded queue

prvok 4
prvok 3
prvok 2

Dátové štruktúry

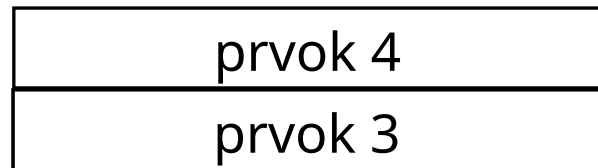
- Ohraničená fronta – bounded queue
- Neohraničená fronta – unbounded queue



 dequeue

Dátové štruktúry

- Ohraničená fronta – bounded queue
- Neohraničená fronta – unbounded queue



Dátové štruktúry

- Jednosmerný zreťazený zoznam – single linked list

```
class Zaznam {
public:
    char meno[21], priezvisko[21], cislo[21];
void vloz() { // metoda v ramci triedy
void vypis() { // metoda v ramci triedy
};

class Prvok_zoznamu {
public:
    Zaznam zaznam;
    Prvok_zoznamu *dalsi = NULL;
void printPrvok_zoznamu() {
void printPriezvisko(char *priezvisko) {
    Prvok_zoznamu* insert() {
};

int main() {
    Prvok_zoznamu *prvy = NULL;
```

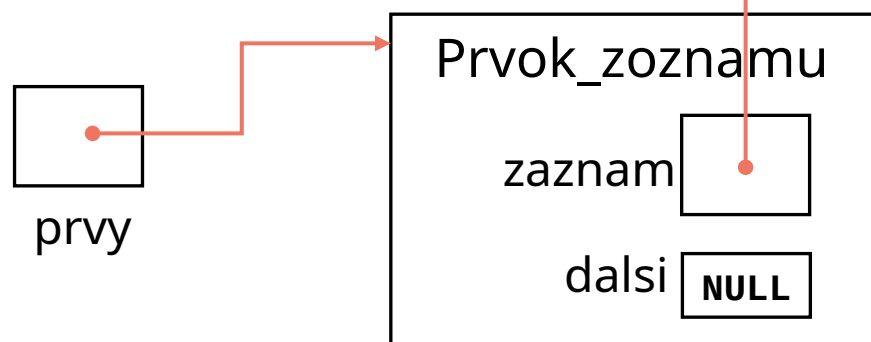
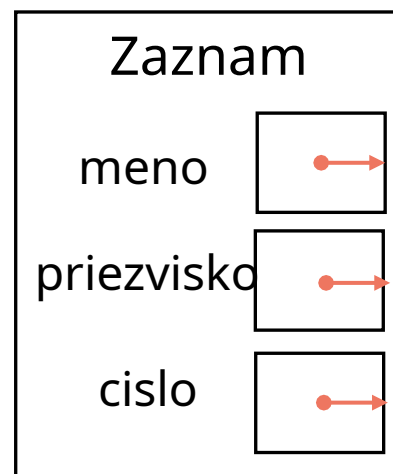


prvy

Dátové štruktúry

- Jednosmerný zreťazený zoznam – single linked list

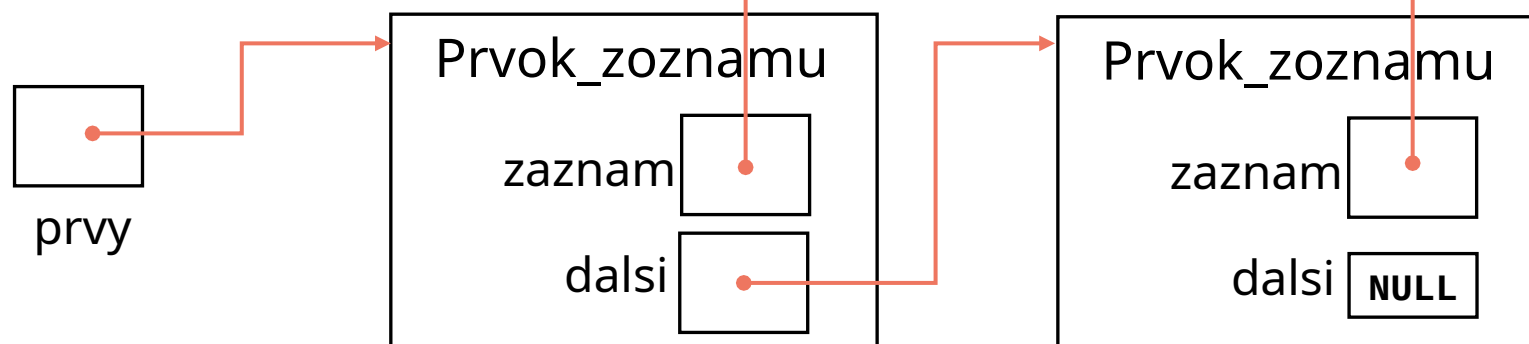
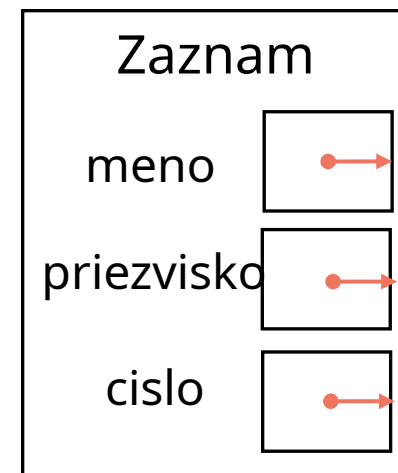
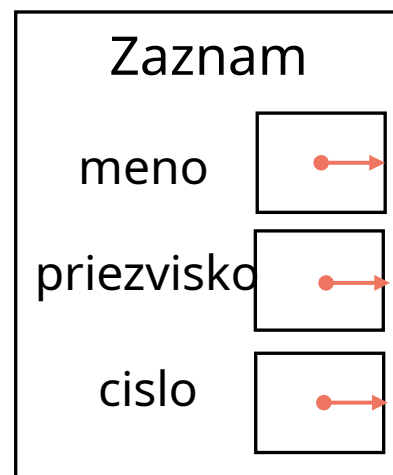
```
class Zaznam {  
    public:  
        char meno[21], priezvisko[21], cislo[21];  
void vloz() { // metoda v ramci triedy  
void vypis() { // metoda v ramci triedy  
};  
  
class Prvok_zoznamu {  
    public:  
        Zaznam zaznam;  
        Prvok_zoznamu *dalsi = NULL;  
void printPrvok_zoznamu() {  
void printPriezvisko(char *priezvisko) {  
        Prvok_zoznamu* insert() {  
};  
  
int main() {  
    Prvok_zoznamu *prvy = NULL;
```



Dátové štruktúry

- Jednosmerný zreťazený zoznam – single linked list

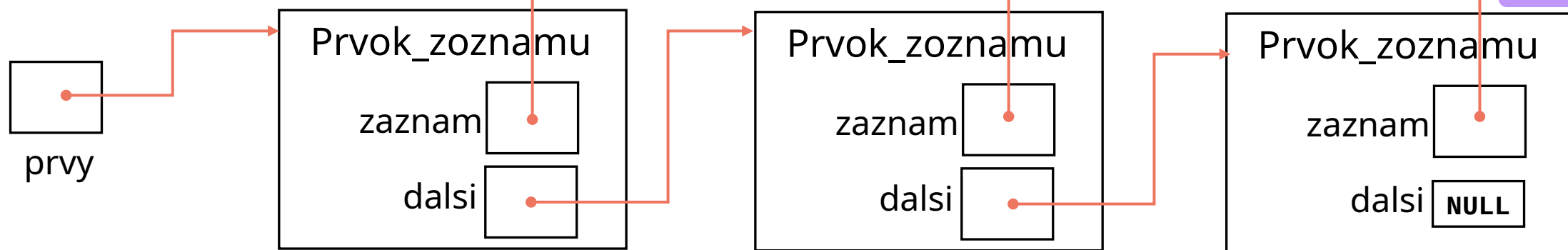
```
class Zaznam {  
    public:  
        char meno[21], priezvisko[21], cislo[21];  
void vloz() { // metoda v ramci triedy  
void vypis() { // metoda v ramci triedy  
};  
  
class Prvok_zoznamu {  
    public:  
        Zaznam zaznam;  
        Prvok_zoznamu *dalsi = NULL;  
void printPrvok_zoznamu() {  
void printPriezvisko(char *priezvisko) {  
        Prvok_zoznamu* insert() {  
};  
  
int main() {  
    Prvok_zoznamu *prvy = NULL;
```



Dátové štruktúry

- Jednosmerný zreťazený zoznam – single linked list

```
class Zaznam {  
public:  
    char meno[21], priezvisko[21], cislo[21];  
void vloz() { // metoda v ramci triedy  
void vypis() { // metoda v ramci triedy  
};  
  
class Prvok_zoznamu {  
public:  
    Zaznam zaznam;  
    Prvok_zoznamu *dalsi = NULL;  
void printPrvok_zoznamu() {  
void printPriezvisko(char *priezvisko) {  
    Prvok_zoznamu* insert() {  
};  
  
int main() {  
    Prvok_zoznamu *prvy = NULL;
```



Dátové štruktúry

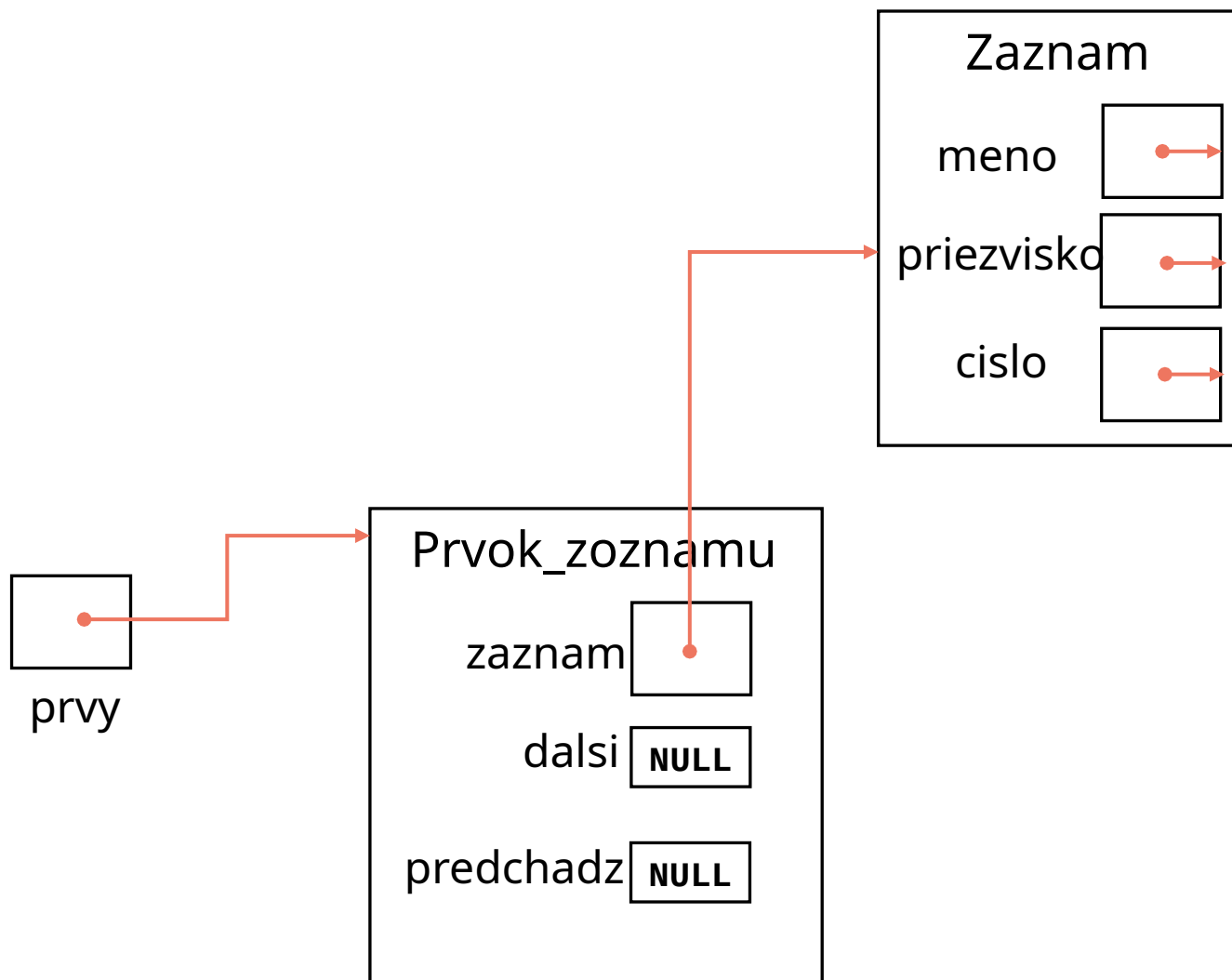
- Obojsmerný zretázený zoznam – doubly linked list



prvy

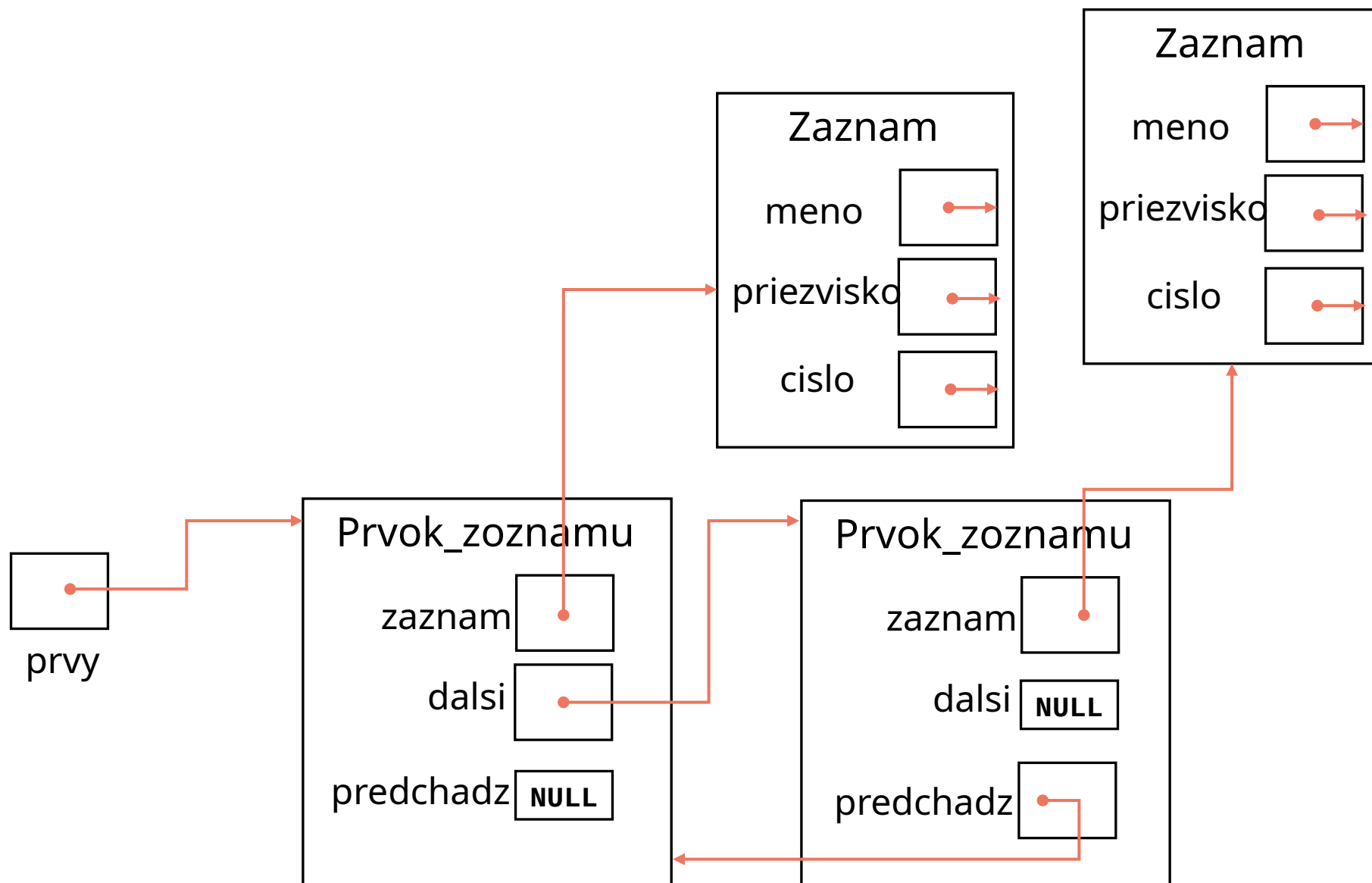
Dátové štruktúry

- Obojsmerný zretázený zoznam – doubly linked list



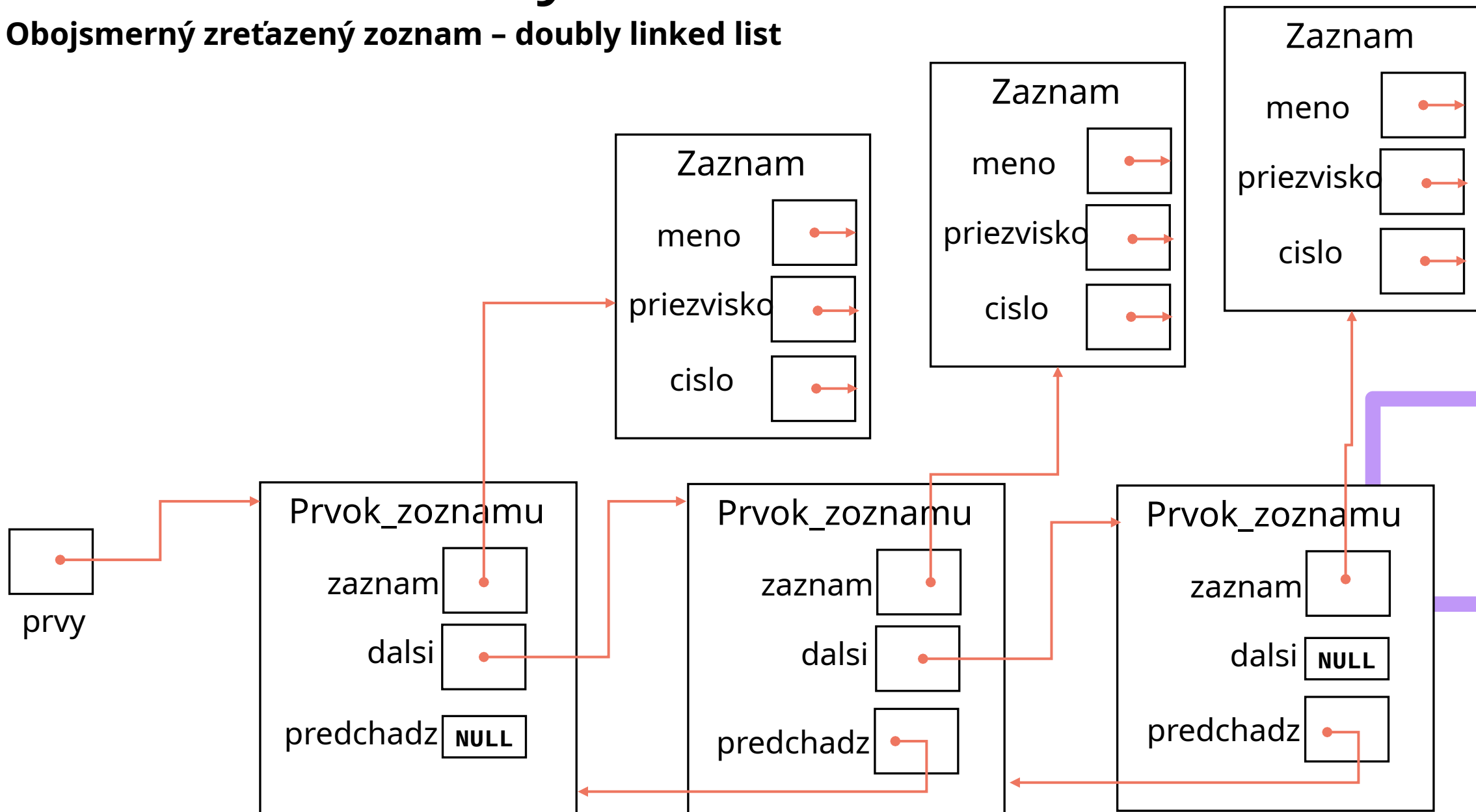
Dátové štruktúry

- Obojsmerný zretázený zoznam – doubly linked list



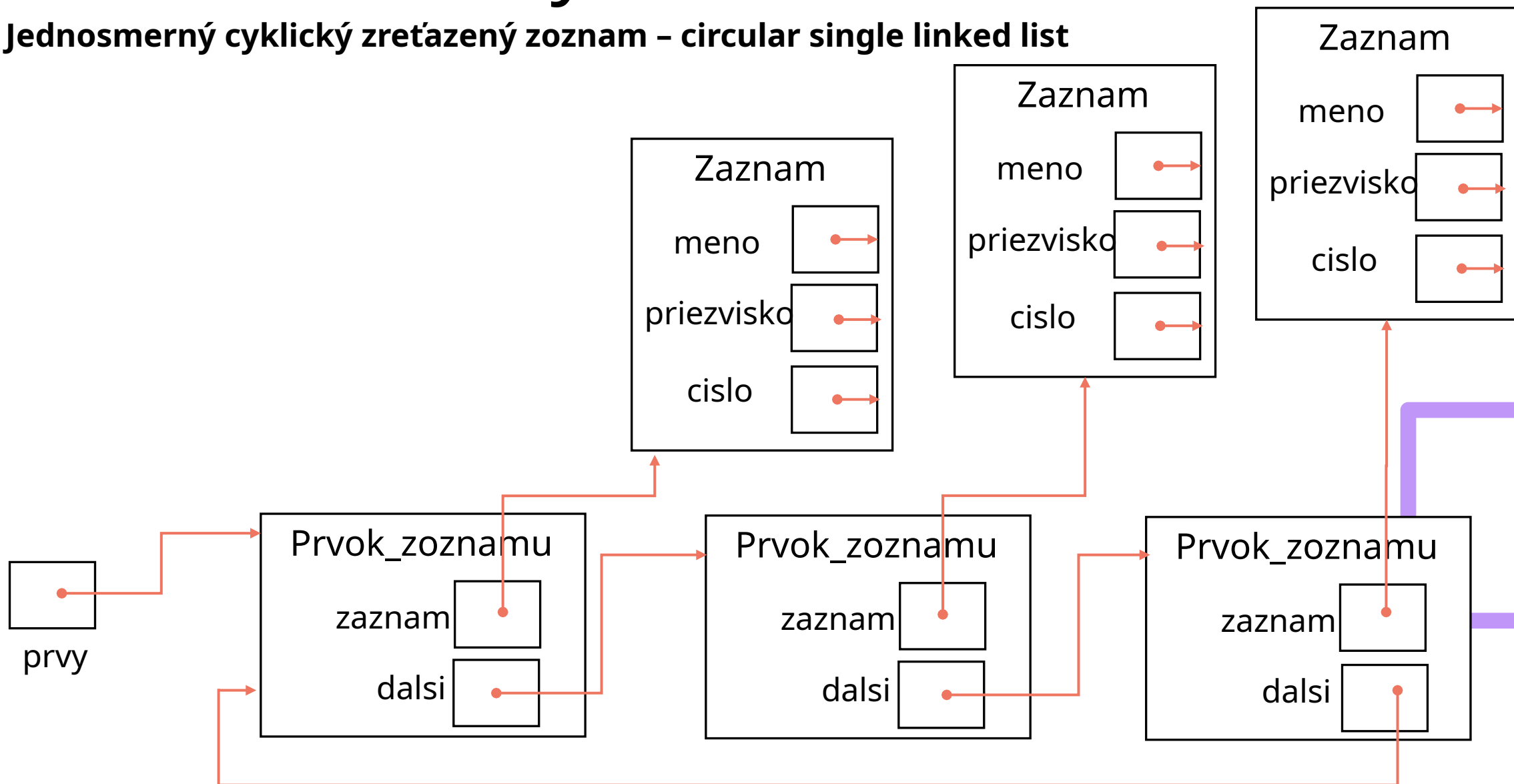
Dátové štruktúry

- Obojsmerný zretázený zoznam – doubly linked list



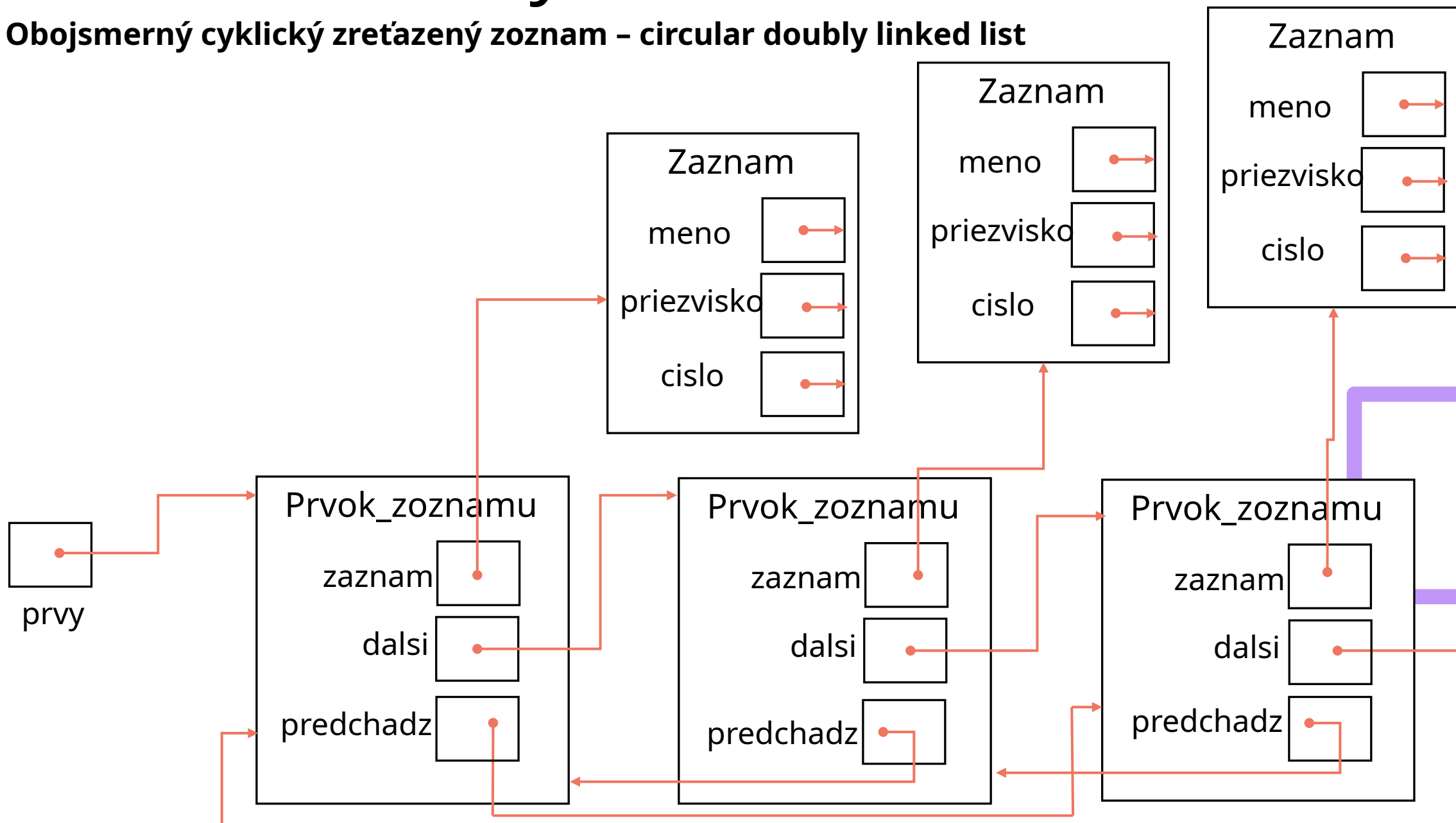
Dátové štruktúry

- Jednosmerný cyklický zreťazený zoznam – circular single linked list



Dátové štruktúry

- Obojsmerný cyklický zretazený zoznam – circular doubly linked list

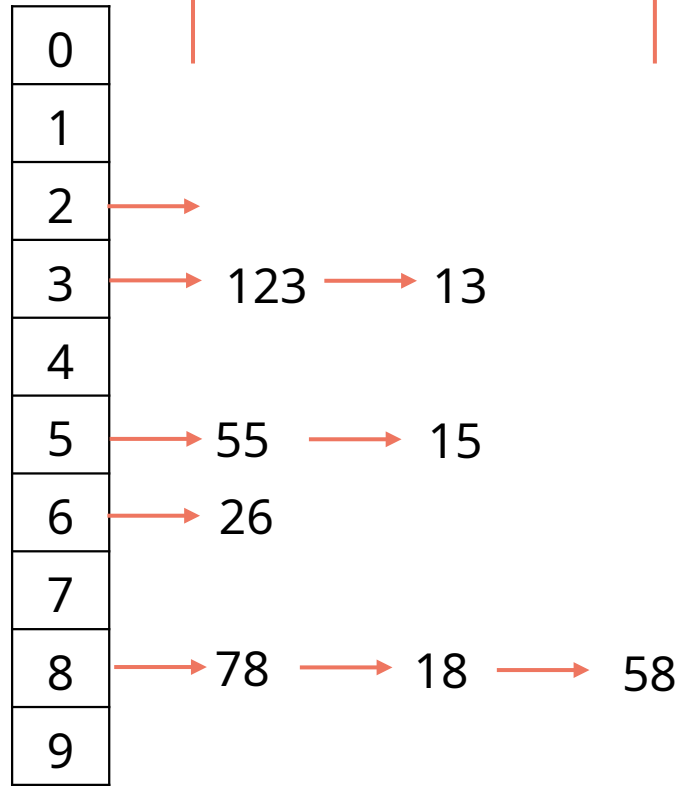


Dátové štruktúry

- Množina – set (a hash set)

{123,
13,
78,
26,
55,
32,
18,
15,
58}

**Hašovacia
funckia, napr.
modulo 10**



Dátové štruktúry

- Mapa – map (a hash table)
- množina párov kľúč, hodnota, (key, value)
- napr studentID, Meno a Priezvisko Index

Keys, napr.
StudentID
{123,
13,
78,
26,
55,
32,
18,
15,
58}

**Hašovacia
funckia, napr.
modulo 10**

Buckets (with separate chaining- overloads - linked lists)

0
1
2
3
4
5
6
7
8
9



123, Ján H. → 13, Peter G.

55, Anna B. → 15, Pavol K.

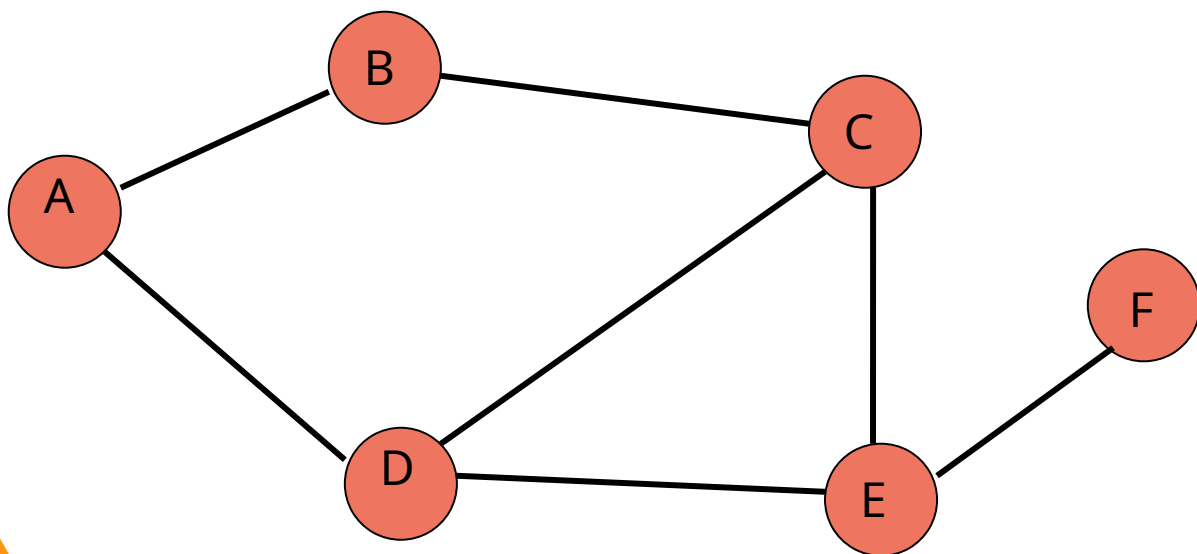
26, Jana V.

78, Adam → 18, Táňa N. → 58, Andrea S.
Z.

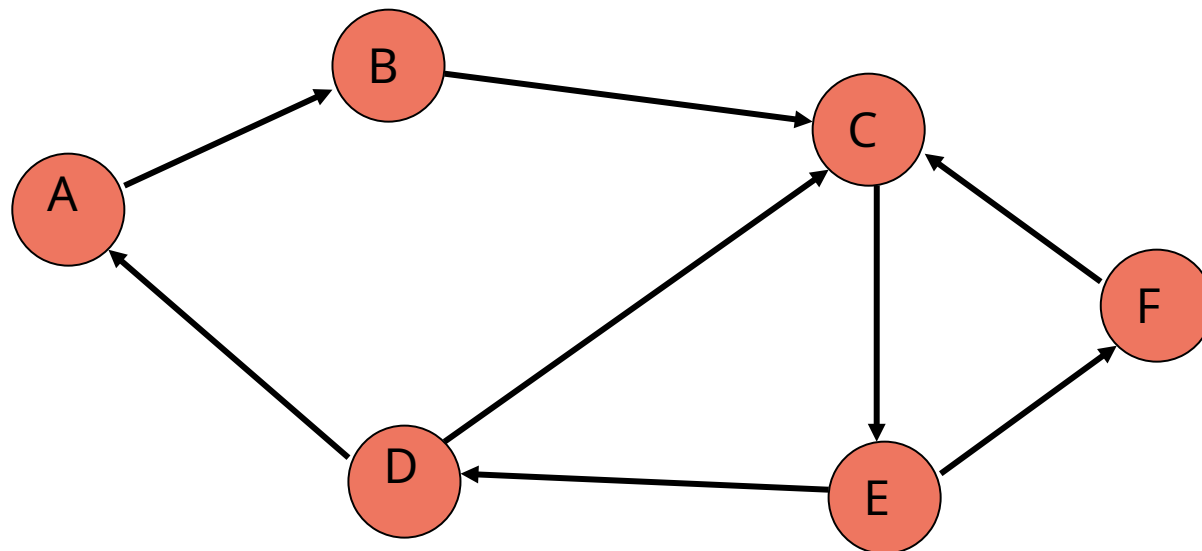
Dátové struktúry

- Graf – graph

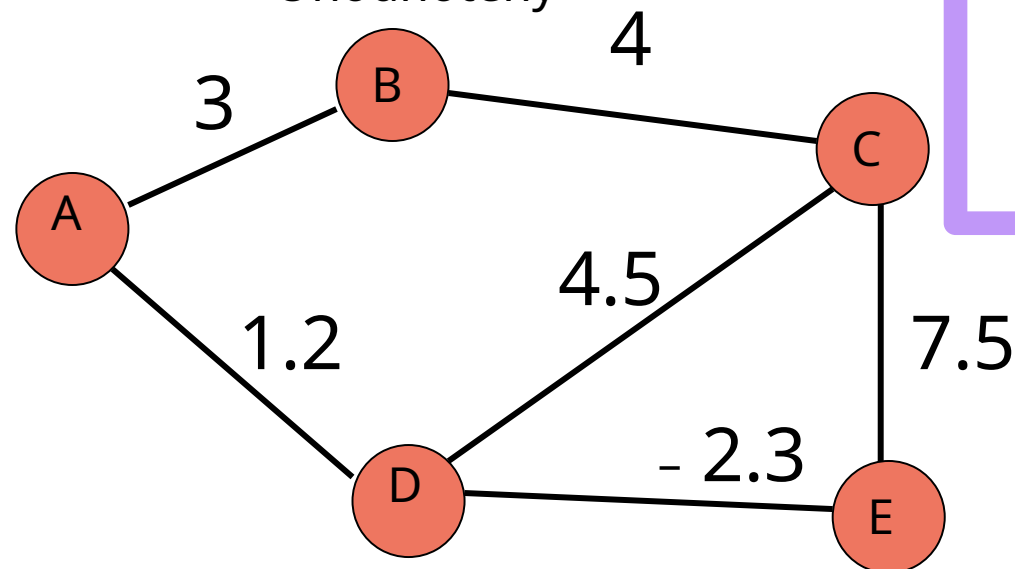
Neorientovaný



Orientovaný



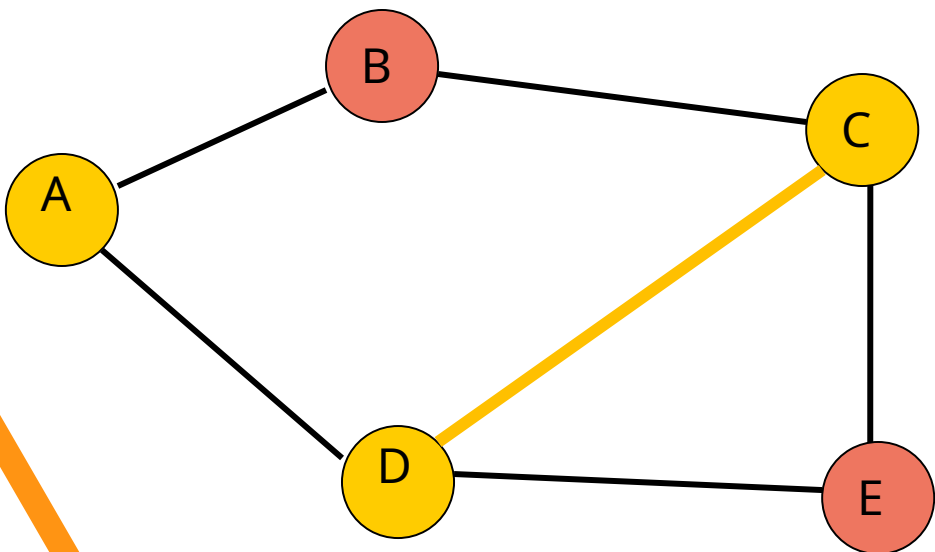
Ohodnotený



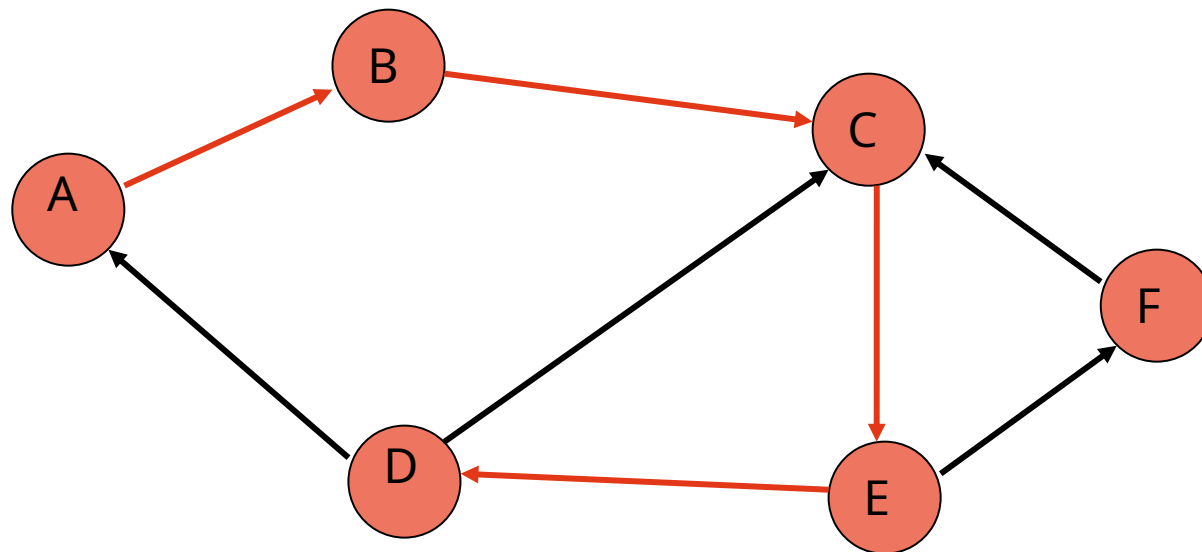
Dátové štruktúry

- Graf – graph

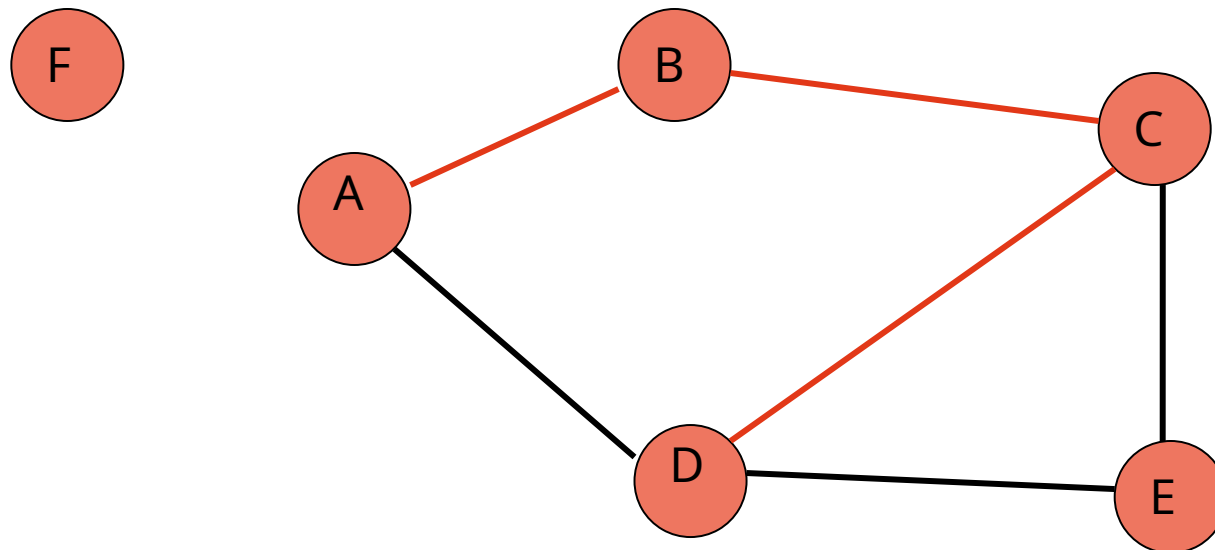
Podgraf



Cesta



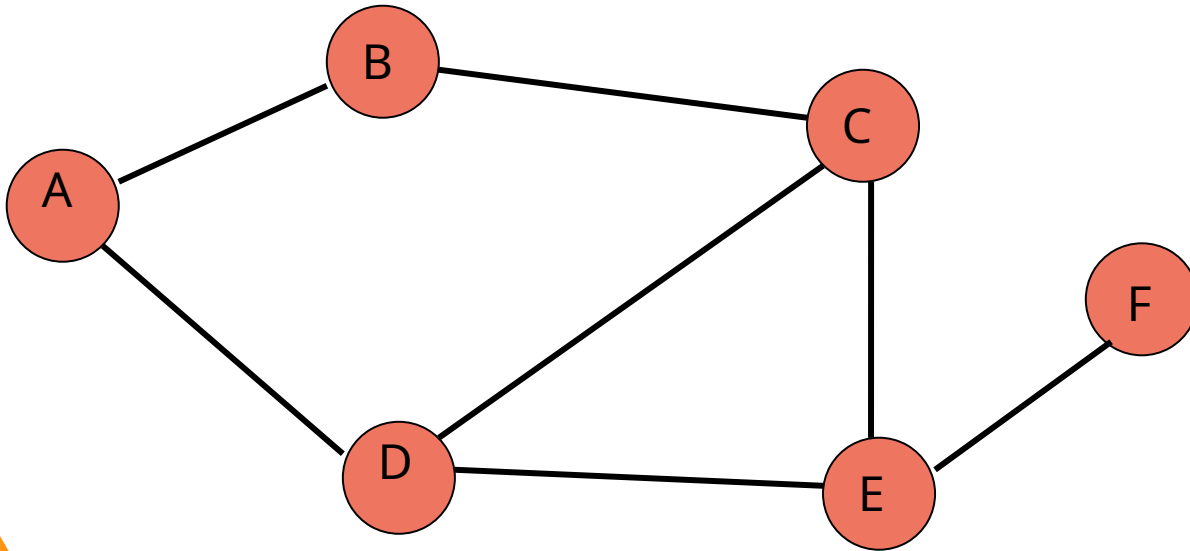
Cesta



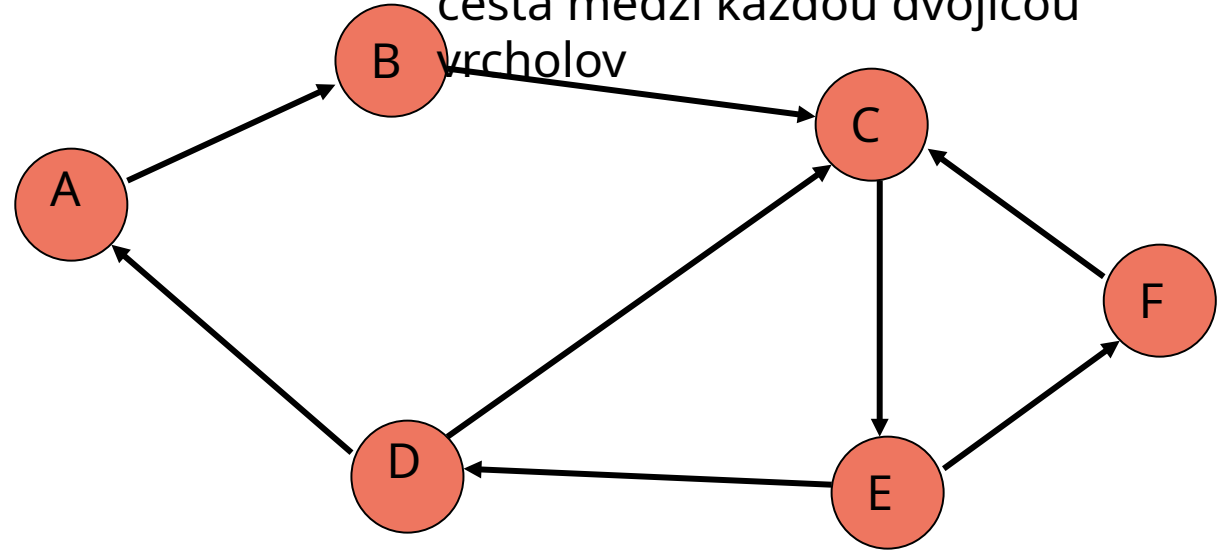
Dátové štruktúry

- Graf – graph

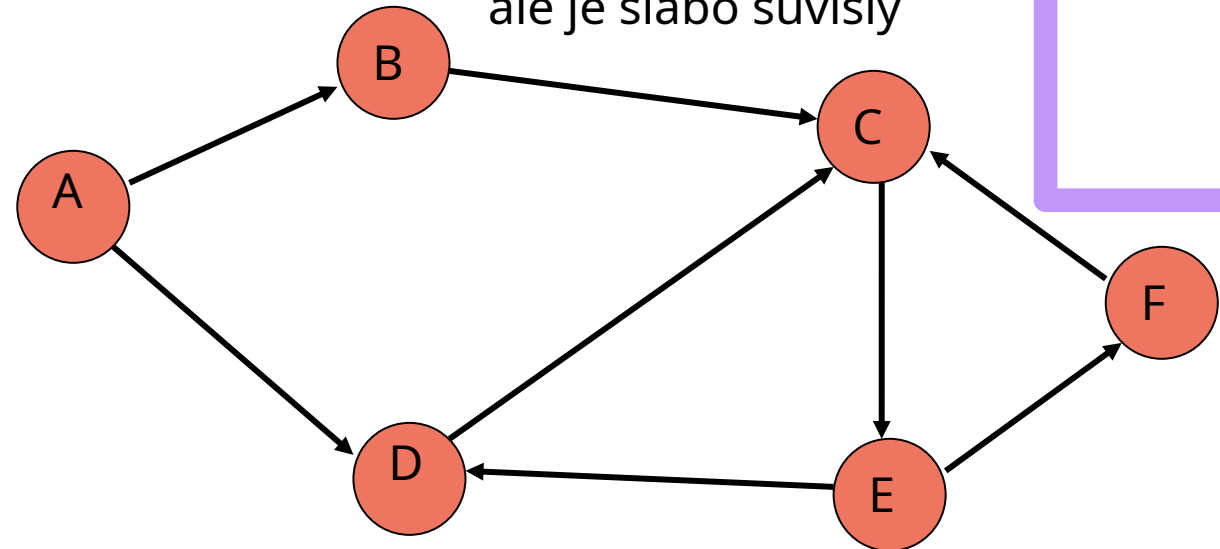
Súvislý – cesta medzi každou dvojicou vrcholov



Silne súvislý – orientovaná cesta medzi každou dvojicou vrcholov



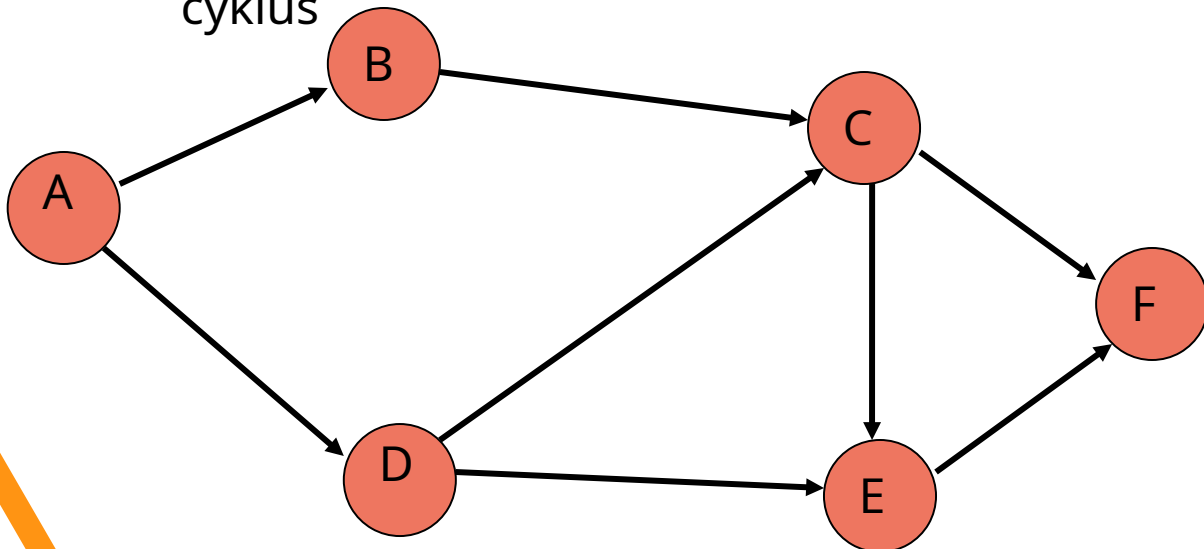
Nie je silne súvislý, ale je slabo súvislý



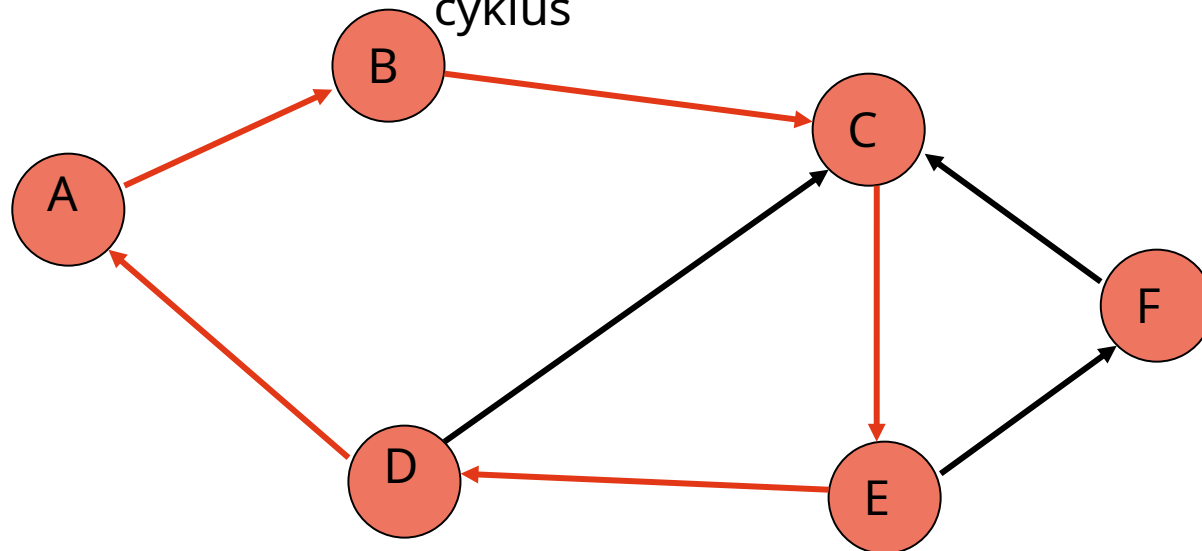
Dátové štruktúry

- Graf – graph

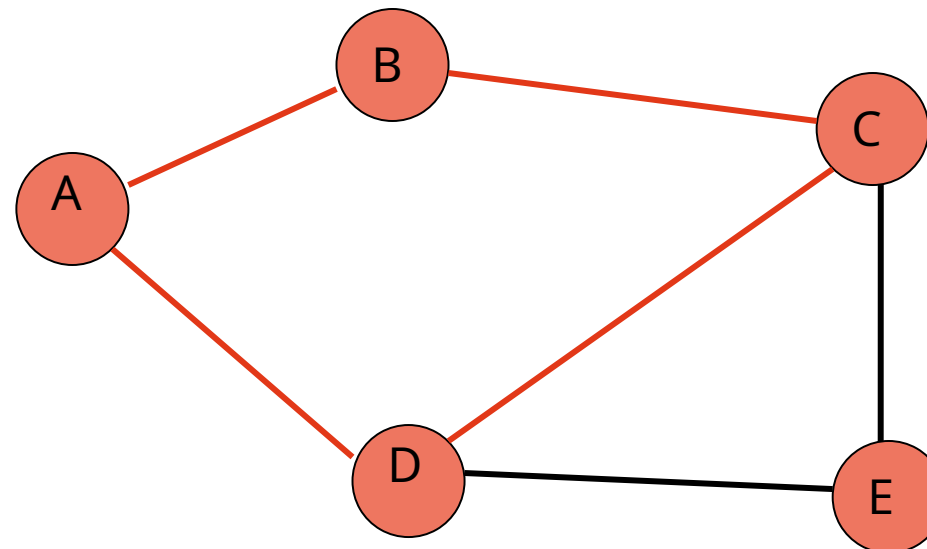
Acyklický = žiadny
cyklus



Orientovaná cesta -
cyklus



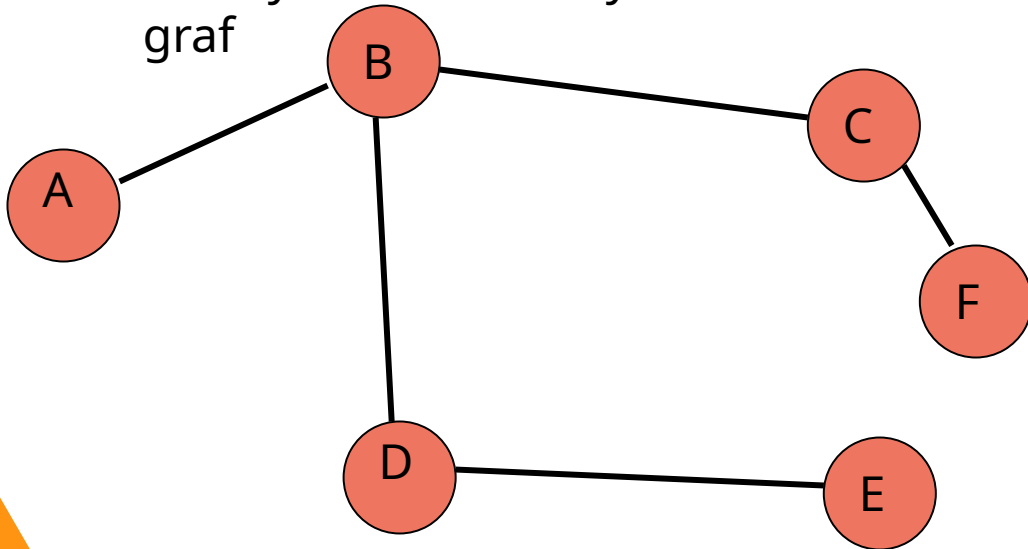
Cesta - cyklus



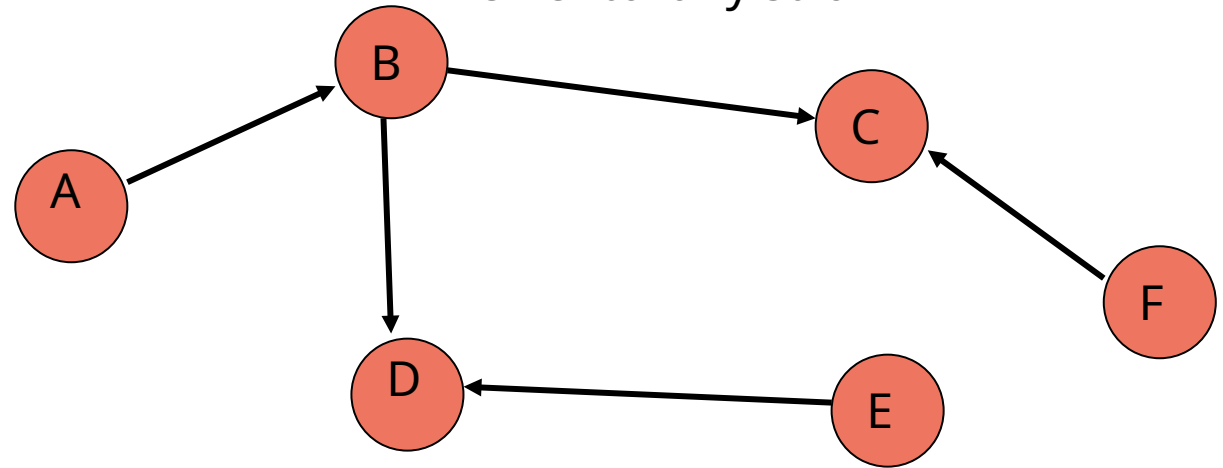
Dátové štruktúry

- Strom – tree

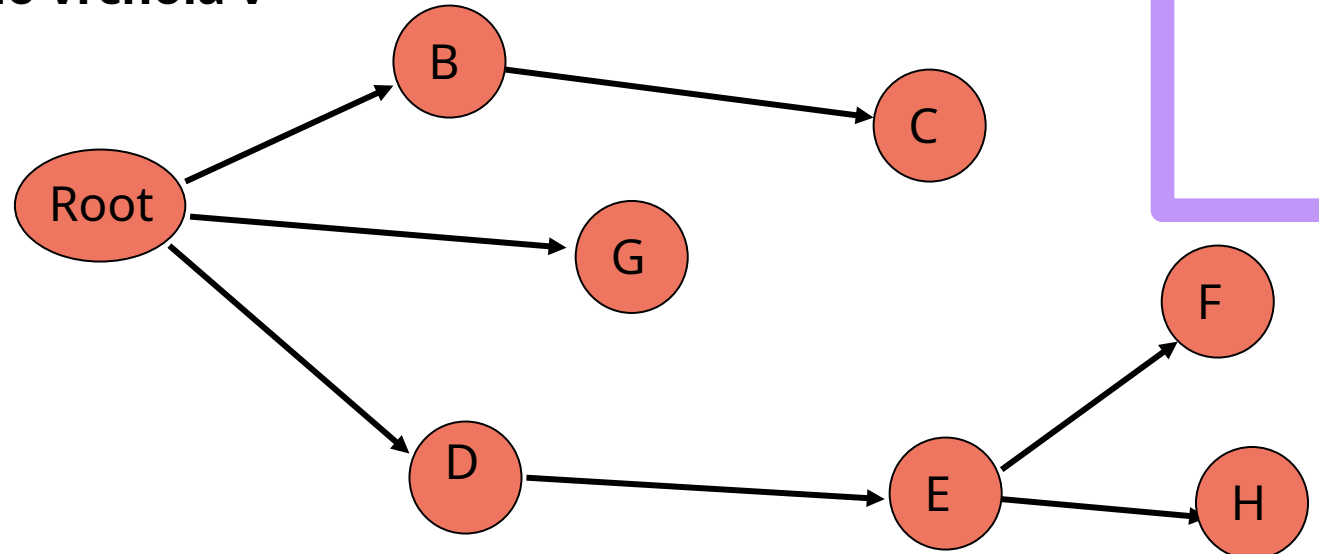
Strom = acyklický,
súvislý neorientovaný
graf



Orientovaný strom

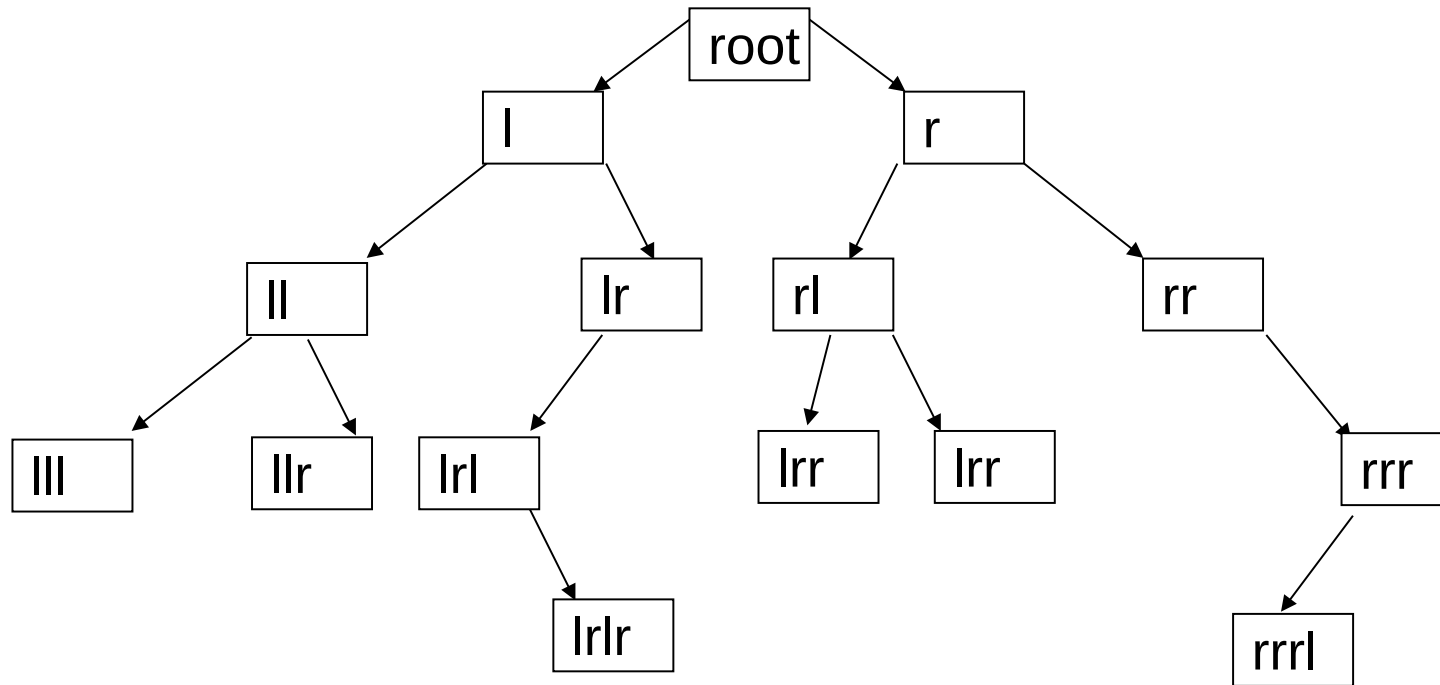


Koreňový orientovaný strom – rooted directed tree – abrorescence = existuje práve jeden vrchol, nazývaný **koreň**, taký, že pre každý **iný vrchol v** existuje **práve jedna cesta z koreňa do vrchola v**



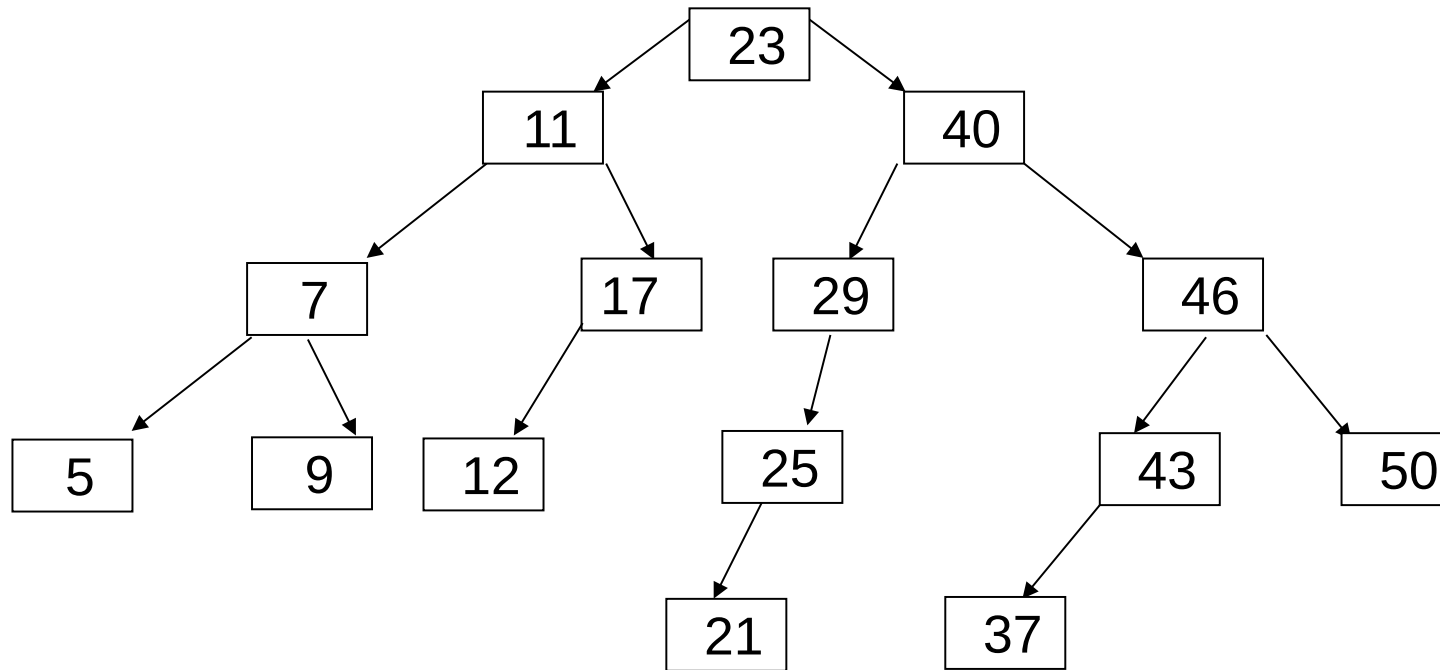
Dátové štruktúry

- **Binárny strom - binary tree = koreňový orientovaný strom, v ktorom každý vrchol má najviac dvoch nasledovníkov**



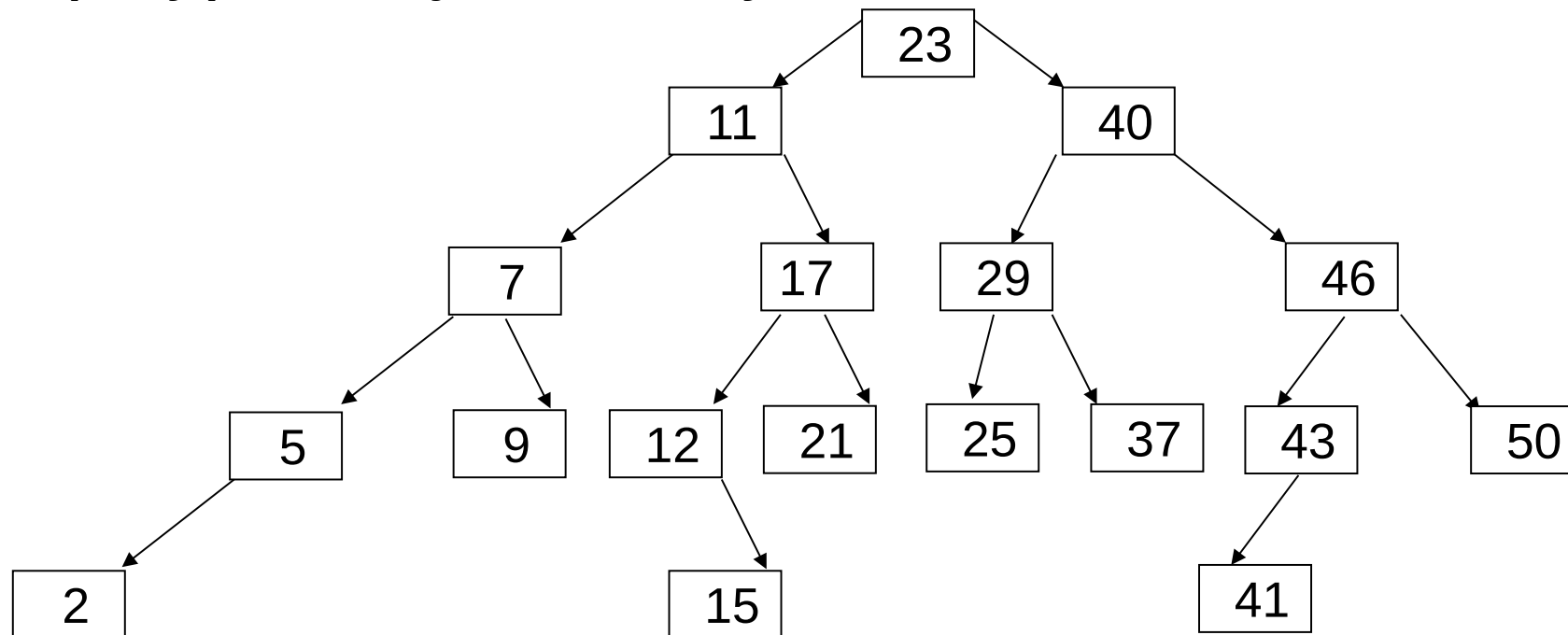
Dátové štruktúry

- Binárny vyhľadávací strom – binary search tree = vrcholy sú usporiadané



Dátové štruktúry

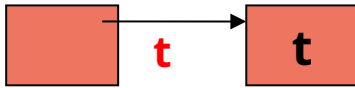
- **Balancovaný binárny strom** - **balanced binary tree** = práve vtedy keď pre každý vrchol platí:
 1. rozdiel výšky jeho ľavého a pravého podstromu je maximálne 1
 2. jeho ľavý podstrom je balancovaný
 3. jeho pravý podstrom je balancovaný



Dátové štruktúry

- Prefixový strom – prefix tree -trie

„**t**he is the most used word there“



Dátové štruktúry

- Prefixový strom – prefix tree -trie

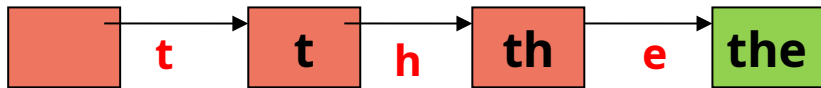
„**th**e is **the** most used word there“



Dátové štruktúry

- Prefixový strom – prefix tree -trie

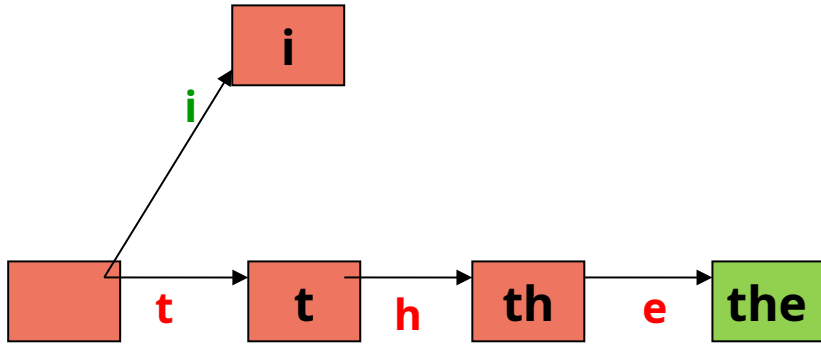
„**the** is **the** most used word there“



Dátové štruktúry

- Prefixový strom – prefix tree -trie

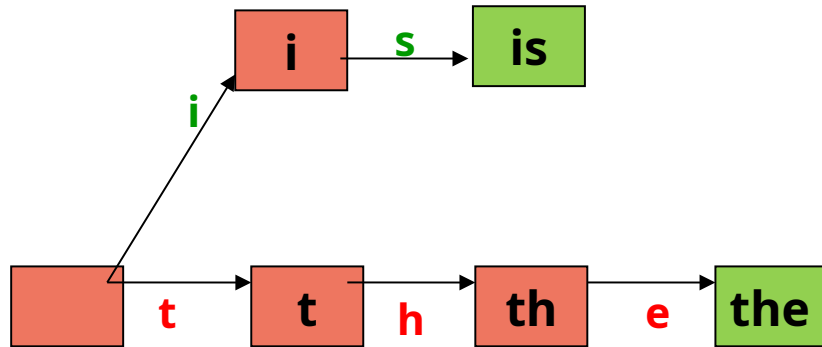
„**the** **i**s **the** most used word there“



Dátové štruktúry

- Prefixový strom – prefix tree -trie

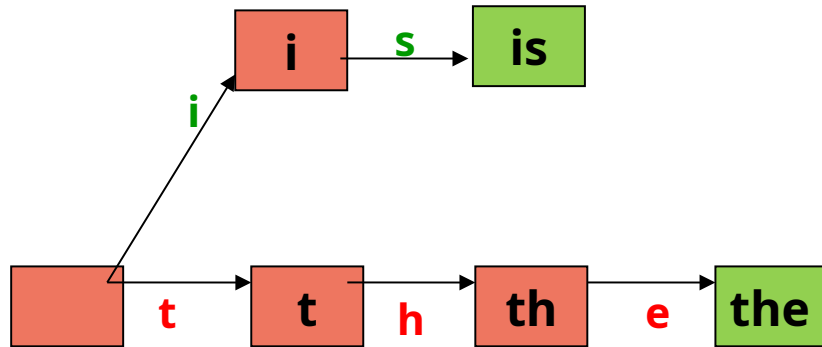
„**the** **is** **the** most used word there“



Dátové štruktúry

- Prefixový strom – prefix tree -trie

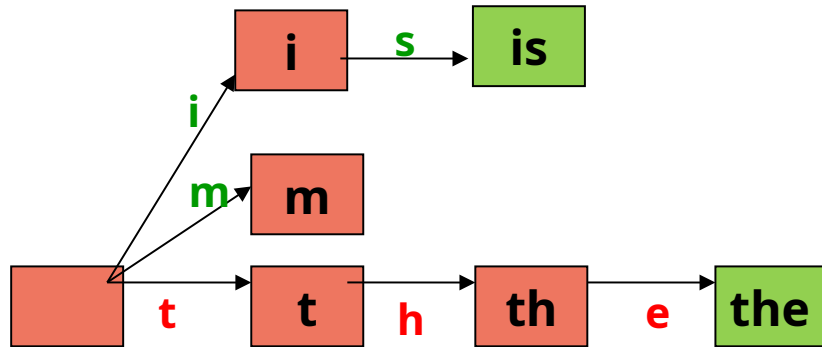
„**the is the** most used word there“



Dátové štruktúry

- Prefixový strom – prefix tree -trie

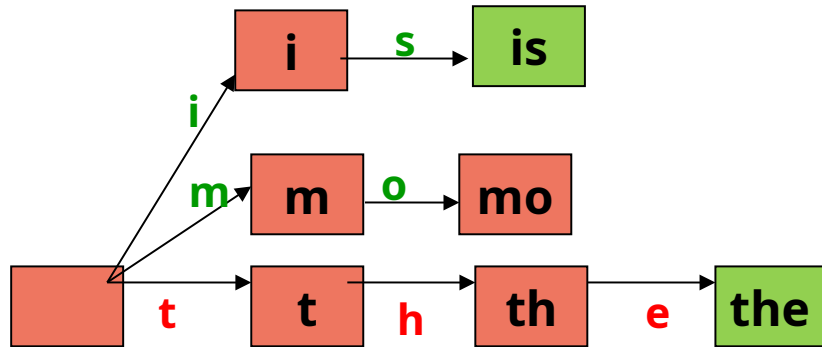
„**the is the m**ost used word there“



Dátové štruktúry

- Prefixový strom – prefix tree -trie

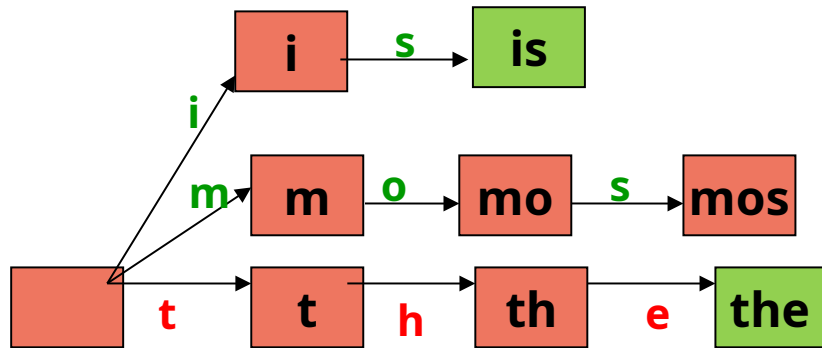
„**the is the mo**st used word there“



Dátové štruktúry

- Prefixový strom – prefix tree -trie

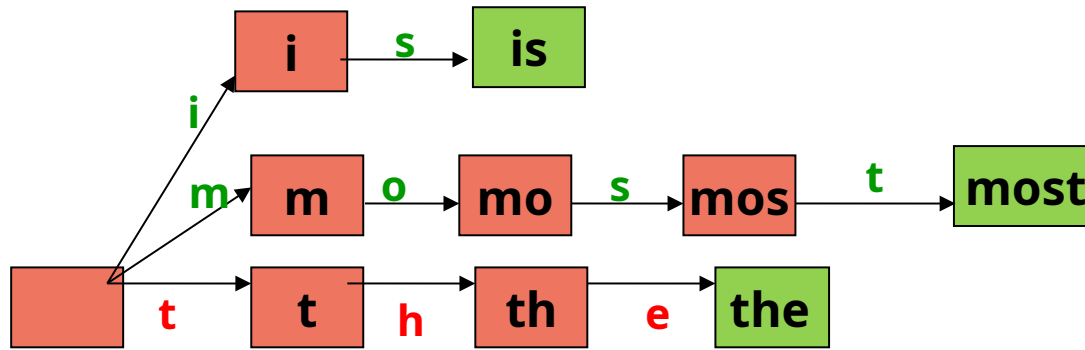
„**the is the mos**t used word there“



Dátové štruktúry

- Prefixový strom – prefix tree -trie

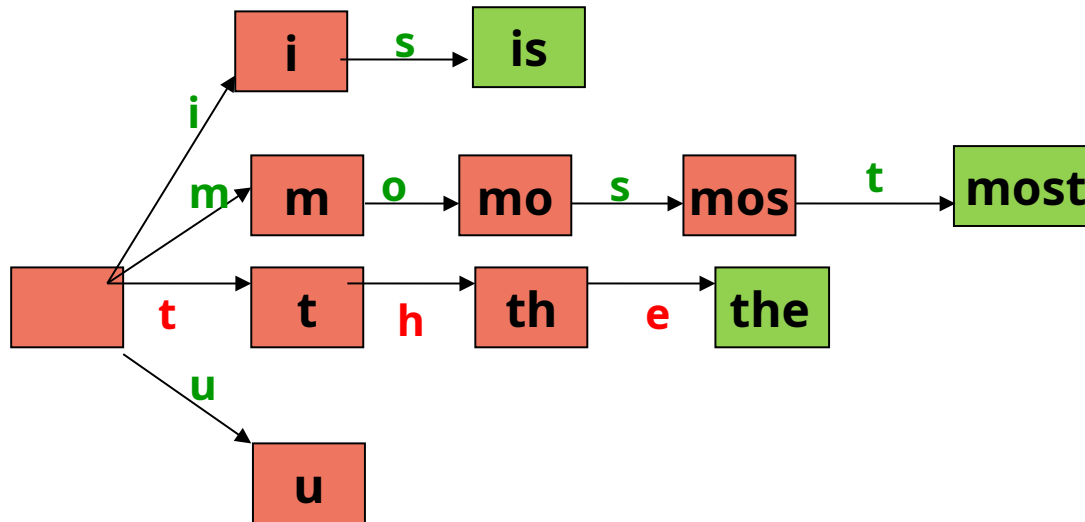
„**the is the most** used word there“



Dátové štruktúry

- Prefixový strom – prefix tree -trie

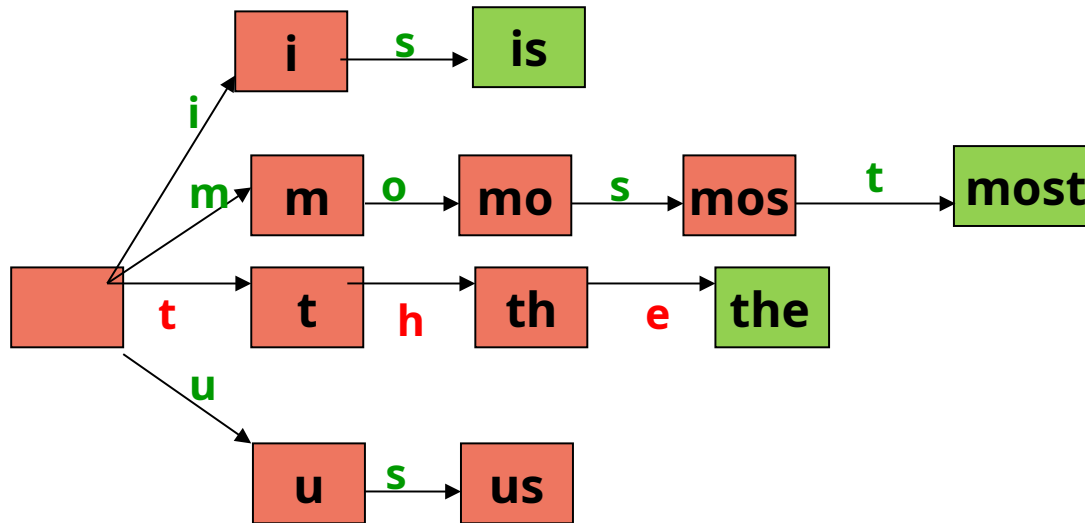
„*the is the most* **u**sed word there“



Dátové štruktúry

- Prefixový strom – prefix tree -trie

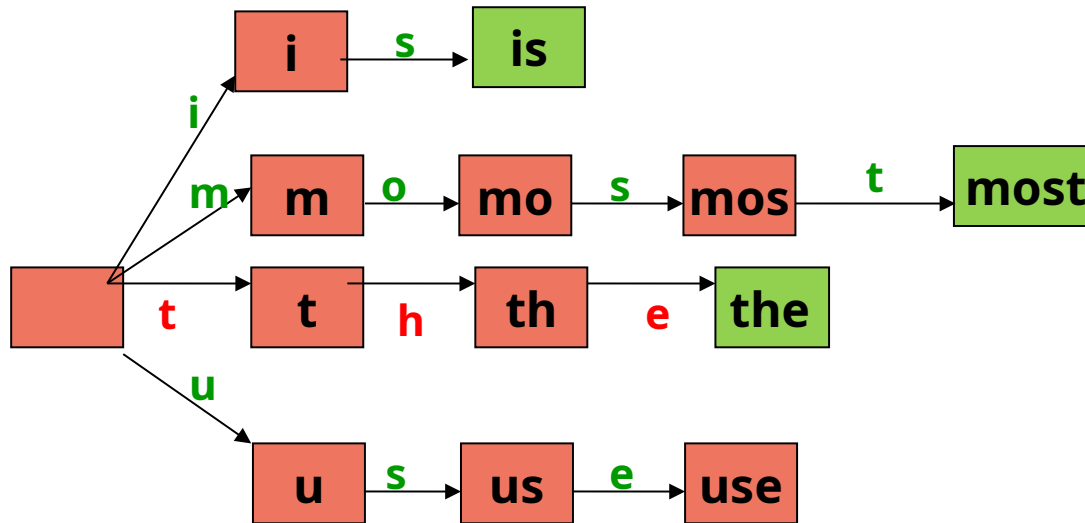
„**the** **is** **the** **most** **us**ed word there“



Dátové štruktúry

- Prefixový strom – prefix tree -trie

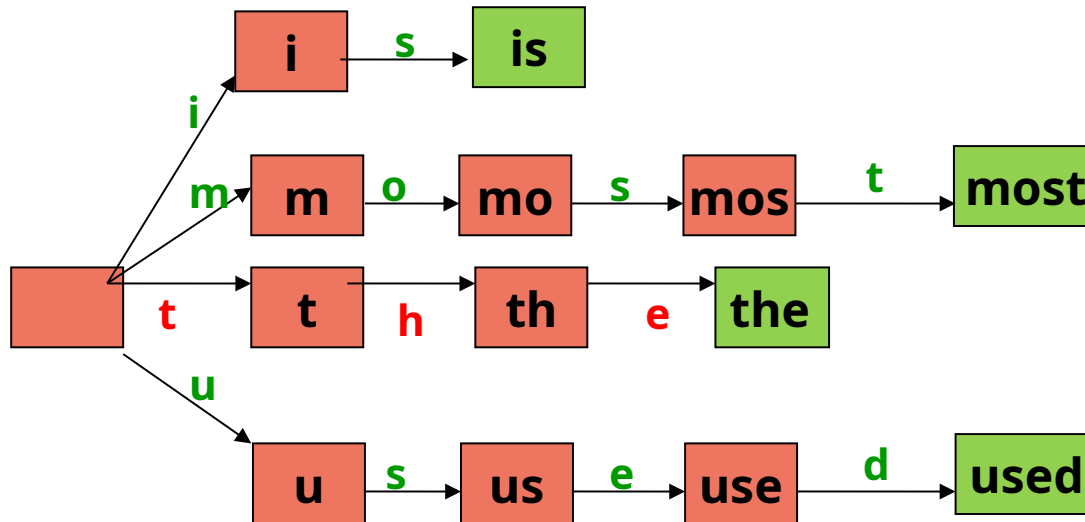
„**the is the most use**d word there“



Dátové štruktúry

- Prefixový strom – prefix tree -trie

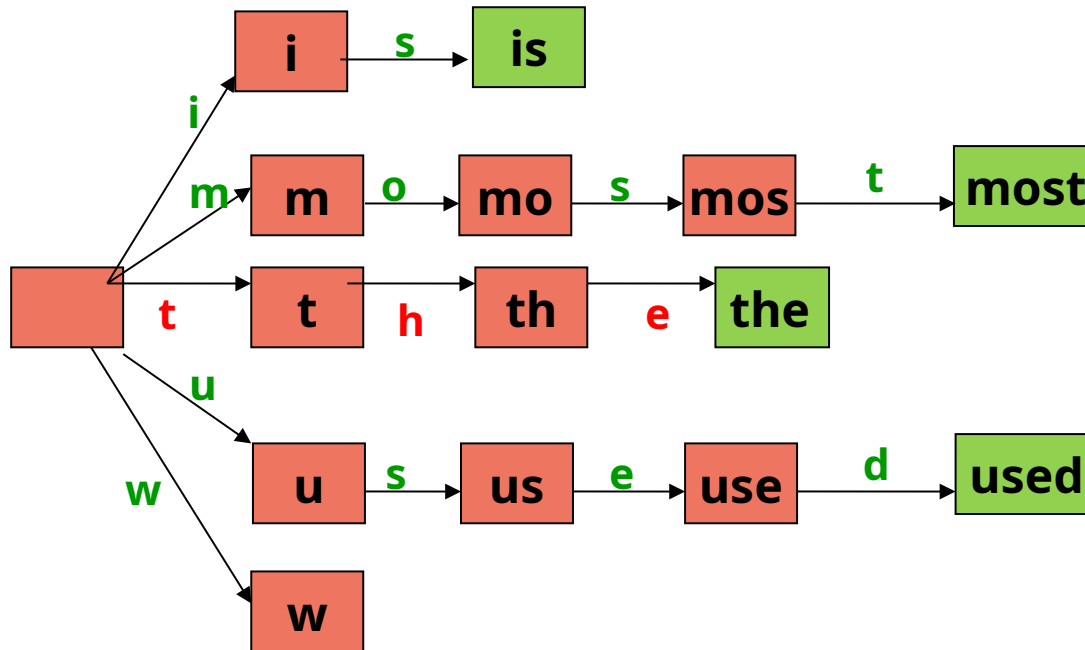
“**the** is **the** most used word there”



Dátové štruktúry

- Prefixový strom – prefix tree -trie

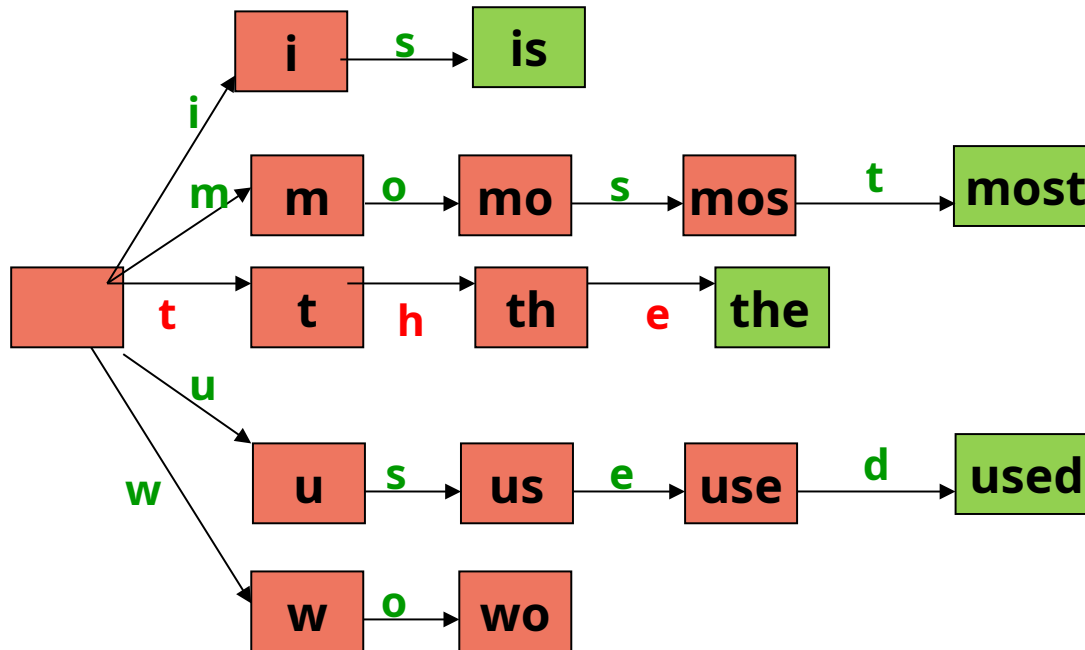
“**the** is **the** most used word there”



Dátové štruktúry

- Prefixový strom – prefix tree -trie

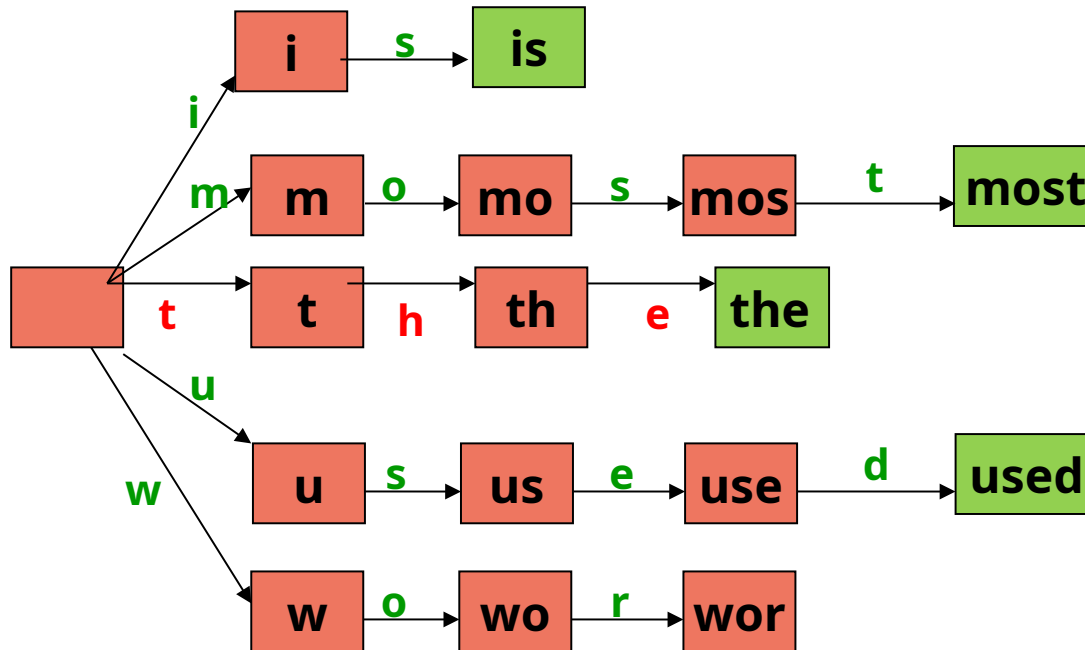
“**the** is **the** most used wo_{rd} there”



Dátové štruktúry

- Prefixový strom – prefix tree -trie

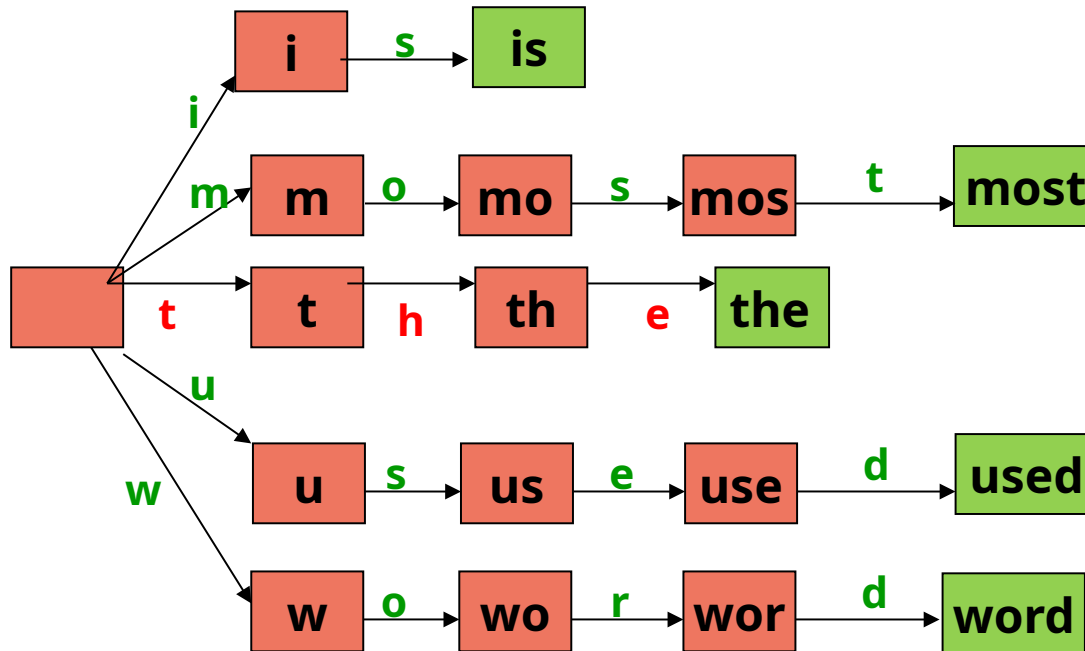
„**the** is **the** most used word there“



Dátové štruktúry

- Prefixový strom – prefix tree -trie

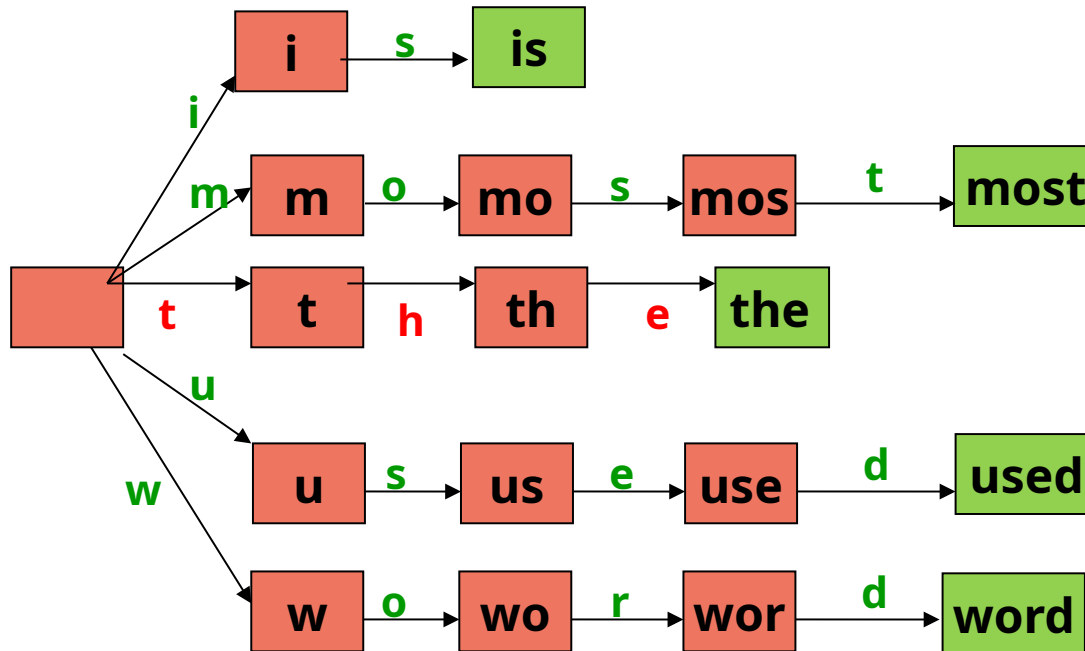
„**the** is **the** most used word there“



Dátové štruktúry

- Prefixový strom – prefix tree -trie

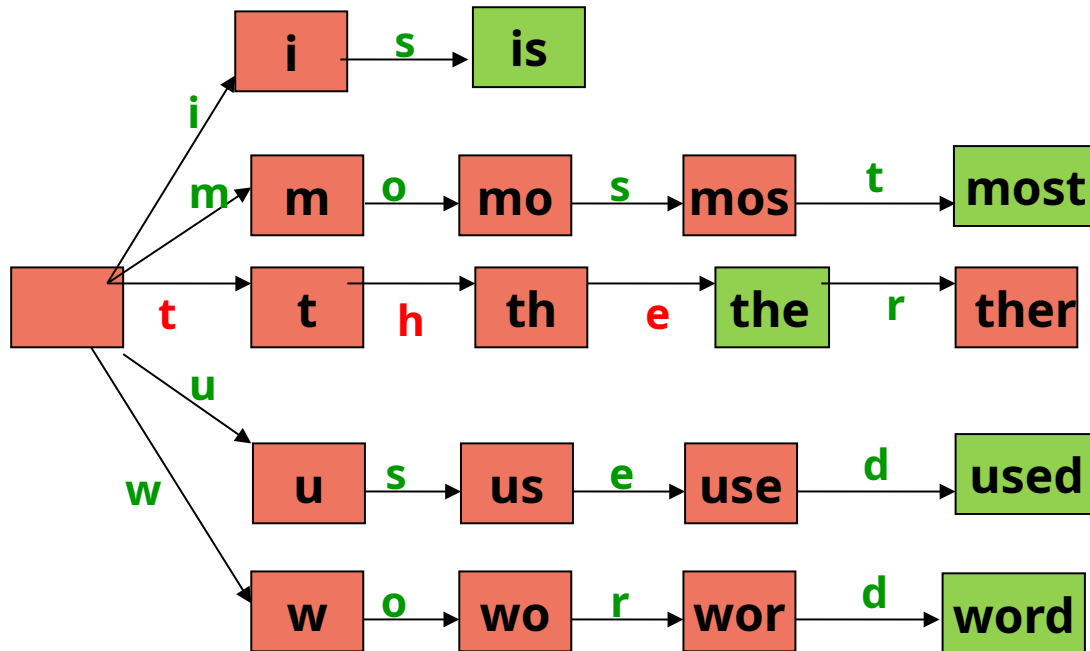
„**the** is **the** most used word **the**re“



Dátové štruktúry

- Prefixový strom – prefix tree -trie

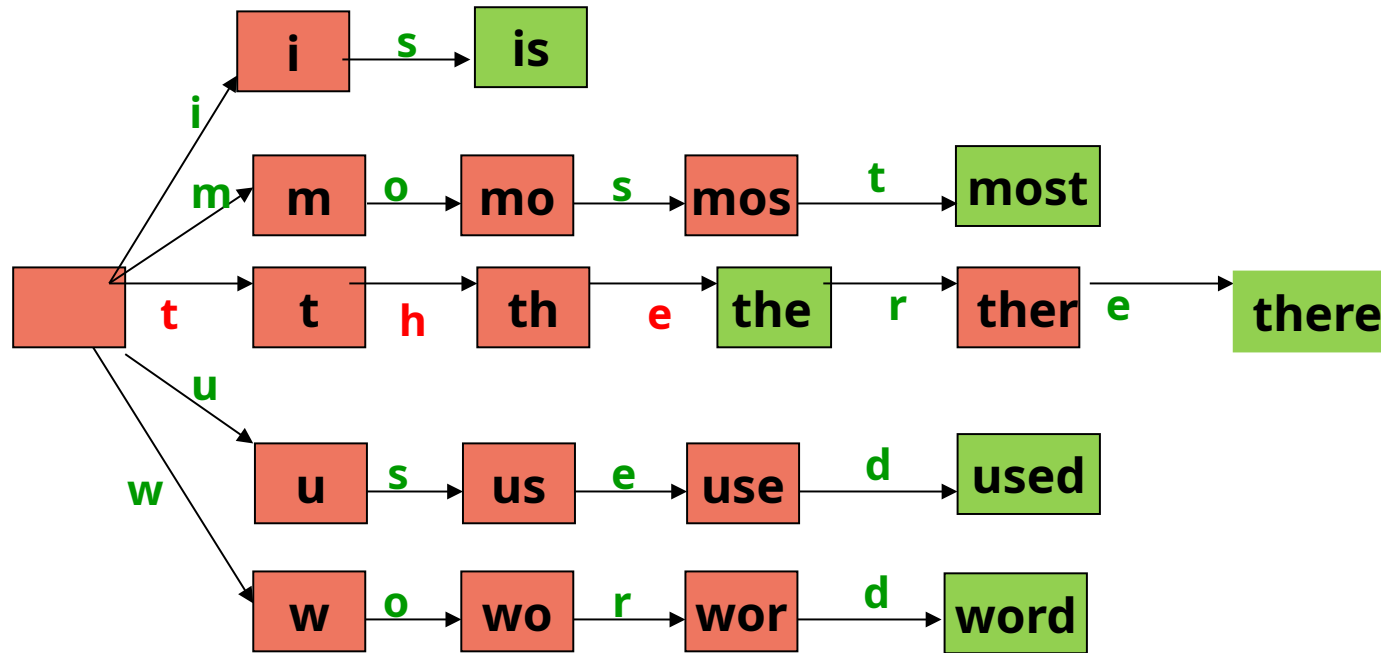
„**the** is **the** most used word there”



Dátové štruktúry

- Prefixový strom – prefix tree -trie

„**the** is **the** most used word there“



Dátové štruktúry

- **Halda a prioritná fronta – heap and priority queue** = pre každý vrchol C , ak P je nadradený uzol C , potom hodnota P je väčšia alebo rovná hodnote C .

