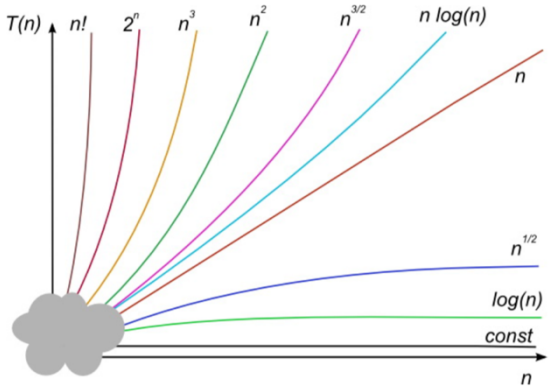
II. Gyakorlat(2019.02.19)

Nevezetes nagyságrendek:

Példák különféle műveletigényű algoritmusokra:

- $\Theta(1)$: verem vagy sor bármely művelete
- $\Theta(\log n)$: bináris keresés, Legendre-algoritmus
- $\Theta(\sqrt{n})$: prímszámteszt
- $\Theta(n)$: lineáris keresés, maximum kiválasztás
- $\Theta(n^2)$: beszúró rendezés, buborék rendezés
- $\Theta(n^3)$: mátrixszorzás
- $\Theta(2^n)$: hanoi tornyai
- $\Theta(n!)$: utazóügynök-probléma

Nagyságrendek szemléltetése:

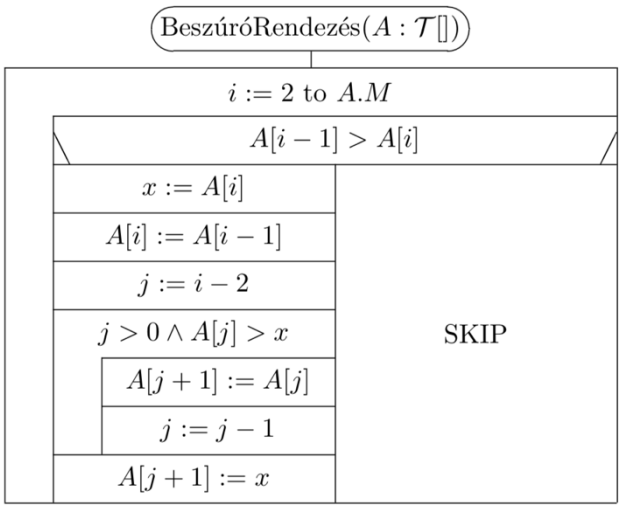


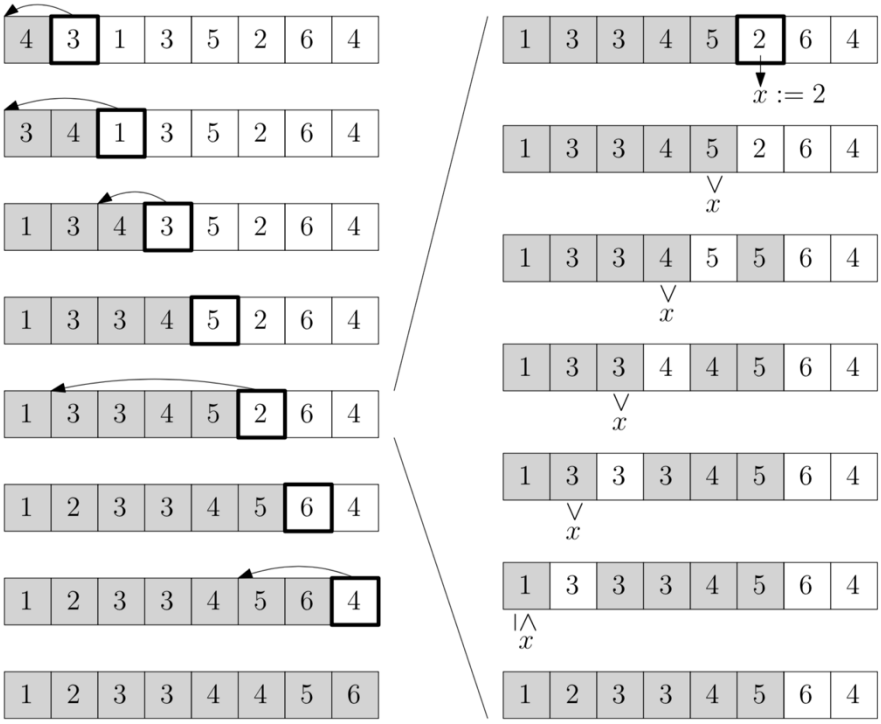
Táblázatos érzékeltetés:

n	log(n)	n^(1/2)	n	n log(n)	n^(3/2)	n^2	n^3	2^n	n!
...									
10	3,32	3,16	10	33,22	31,62	100	1000	1024	3628800
11	3,46	3,32	11	38,05	36,48	121	1331	2048	39916800
12	3,58	3,46	12	43,02	41,57	144	1728	4096	479001600
13	3,70	3,61	13	48,11	46,87	169	2197	8192	6,227E+09
...									
29	4,86	5,39	29	140,88	156,17	841	24389	536870912	8,842E+30
30	4,91	5,48	30	147,21	164,32	900	27000	1,074E+09	2,653E+32
31	4,95	5,57	31	153,58	172,60	961	29791	2,147E+09	8,223E+33
32	5	5,66	32	160	181,02	1024	32768	4,295E+09	2,631E+35
...									
64	6	8	64	384	512	4096	262144	1,845E+19	1,269E+89
...									
128	7	11,31	128	896	1448,15	16384	2097152	3,403E+38	3,86E+215
...									
256	8	16	256	2048	4096	65536	16777216	1,158E+77	****
...									
512	9	22,63	512	4608	11585,24	262144	134217728	1,34E+154	****
...									
1024	10	32	1024	10240	32768	1048576	1,074E+09	****	****

Forrás: Fekete István jegyzete: *https://people.inf.elte.hu/fekete/algoritmusok_jegyzet/*

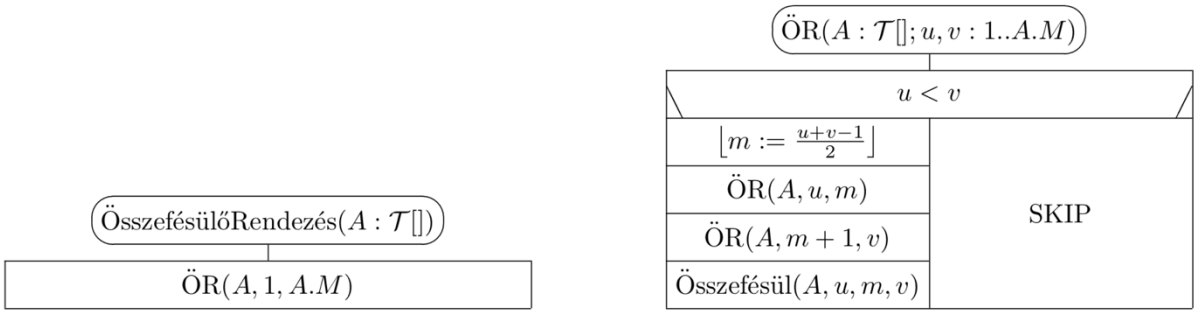
Beszúró rendezés:

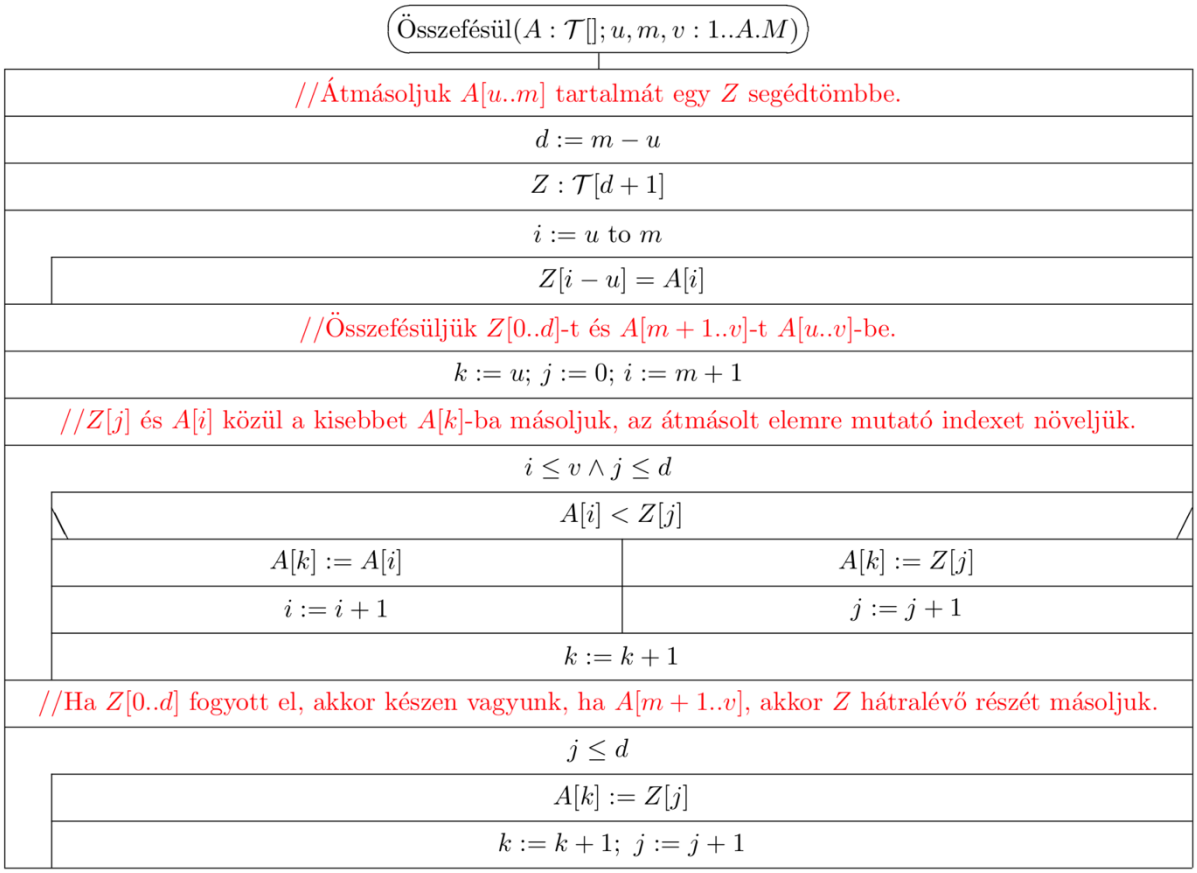


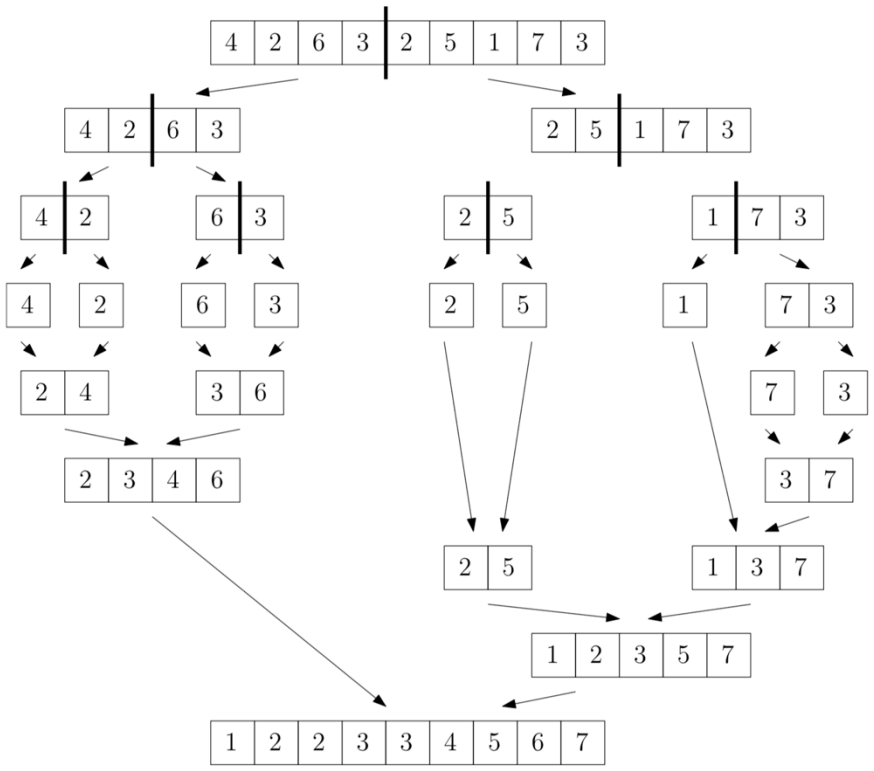


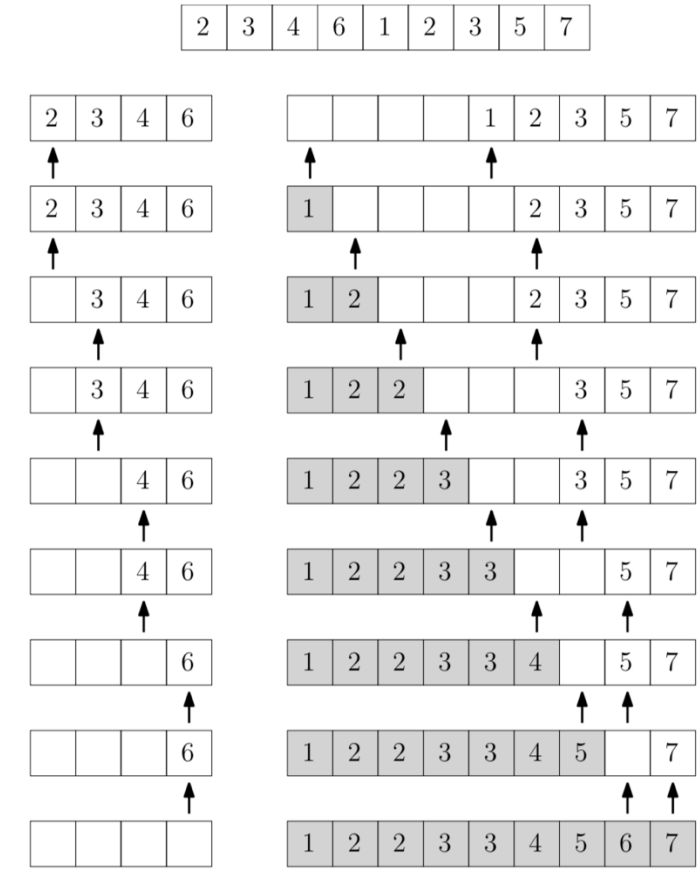
Stabilitás: egy rendező eljárást stabilnak nevezünk, ha nem változtatja meg az egyenlő kulcsú elemek egymáshoz viszonyított sorrendjét. A beszúró rendezés stabil. A buborék rendezés is stabil, viszony a maximum kiválasztásos rendezés nem. Hogyan lehet azzá tenni? Ha a maximum keresése közben az aktuális maximummal egyenlő kulcsú elemet találunk, akkor frissítsük az *ind* változó értékét.

Összefésülő rendezés:





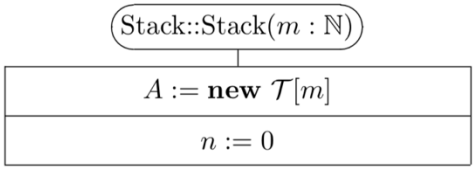

4 2 6 3 2 5 1 7 3
4 2 6 3
2 5 1 7 3
4 2
6 3
2 5
1 7 3
4
2
6
3
2
5
1
7 3
2 4
3 6
7
3
2 3 4 6
3 7
2 5
1 3 7
1 2 3 5 7
1 2 2 3 3 4 5 6 7

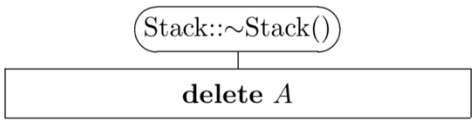

2 3 4 6 1 2 3 5 7

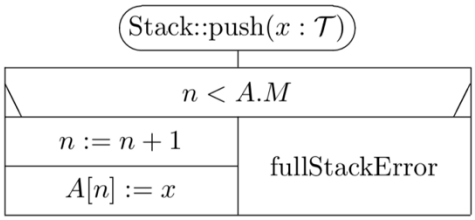
Verem:

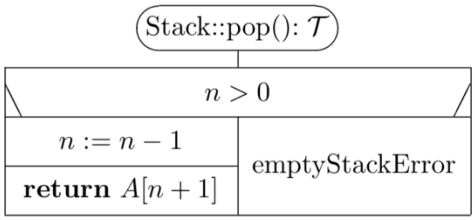
Stack
- $A : \mathcal{T}[]$ - $n : 0..A.M$ //A verem aktuális mérete

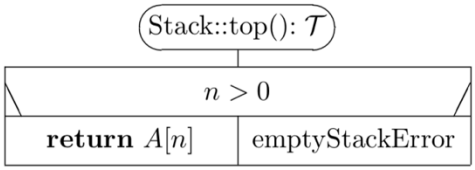
+ Stack($m : \mathbb{N}$) //Konstruktor, létrehoz egy üres vermet.
+ ~Stack() //Destruktor, törli a vermet.
+ push($x : \mathcal{T}$) //x beszúrása a verem tetejére.
+ pop() : $\mathcal{T}$ //Törli, és egyben visszaadja a verem legfelső elemét.
+ top() : $\mathcal{T}$ //A verem legfelső eleme.
+ isFull() : $\mathbb{B}$ //Megadja, hogy tele van-e a verem.
+ isEmpty() : $\mathbb{B}$ //Megadja, hogy üres-e a verem.
+ setEmpty() //Kiüríti a vermet.

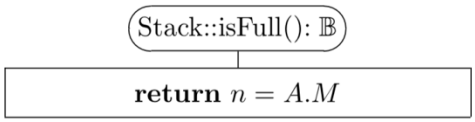


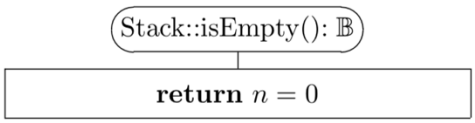


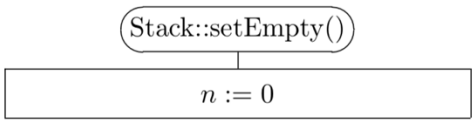


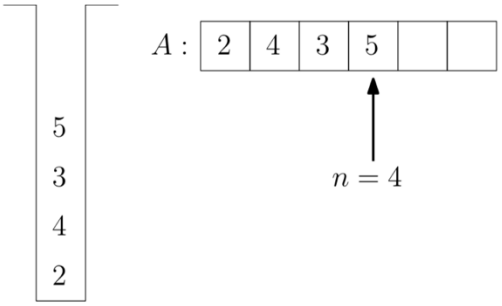












Kitérő: UML – Unified Modeling Language
(https://hu.wikipedia.org/wiki/Unified_Modeling_Language)

A fentebb látható UML diagram alapján a **Stack** osztály a következő képpen nézne ki egy programozási nyelvben leimplementálva. (esetünkben C++-ban). *T* egy tetszőleges típus, ún. sablon típus vagy template argumentum.

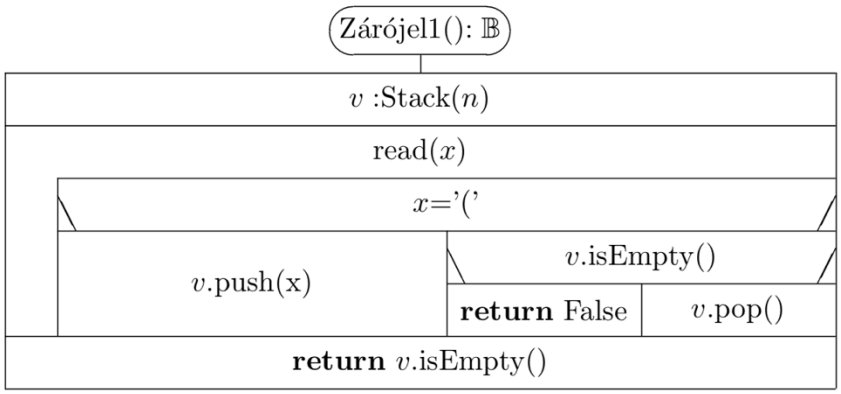
```
template<typename T>
class Stack
{
private:
    T A[];
    int n;

public:
    Stack(int m);
    ~Stack();

    void push(T x);
    T pop();
    T top();
    bool isFull();
    bool isEmplty();
    void setEmpty();
};
```

Veremmel megoldható feladatok:

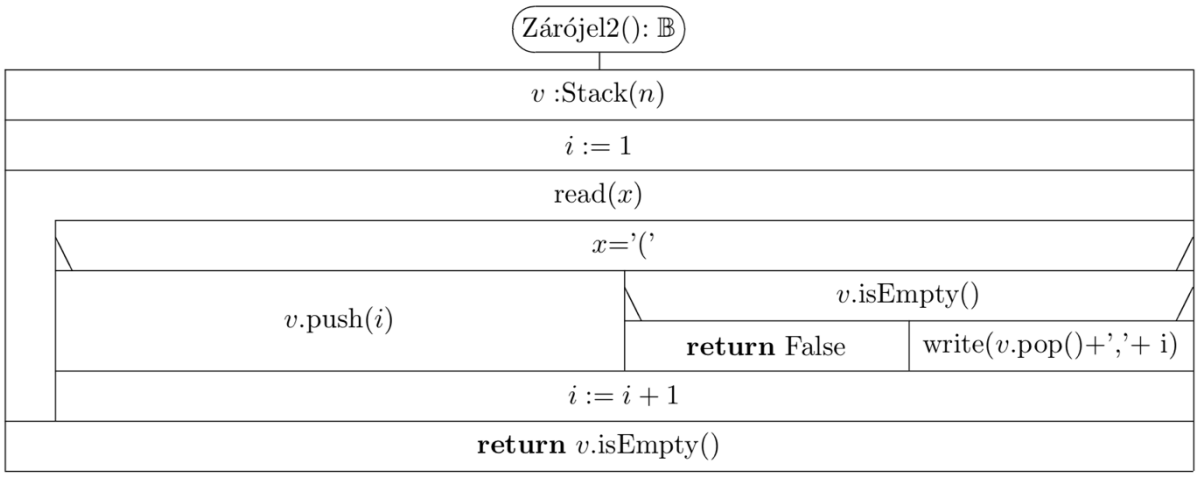
1. Adott egy zárójelekből álló, legfeljebb *n* hosszú karaktersorozat a bemeneten. Olvassuk be, és döntsük el róla, hogy helyes zárójelezést határoz-e meg! Vagyis, hogy párba állíthatók-e a zárójelek úgy, hogy minden nyitó zárójelnek egy olyan csukó zárójel párja van, amely később következik a sorozatban.
   Megoldás: verem segítségével



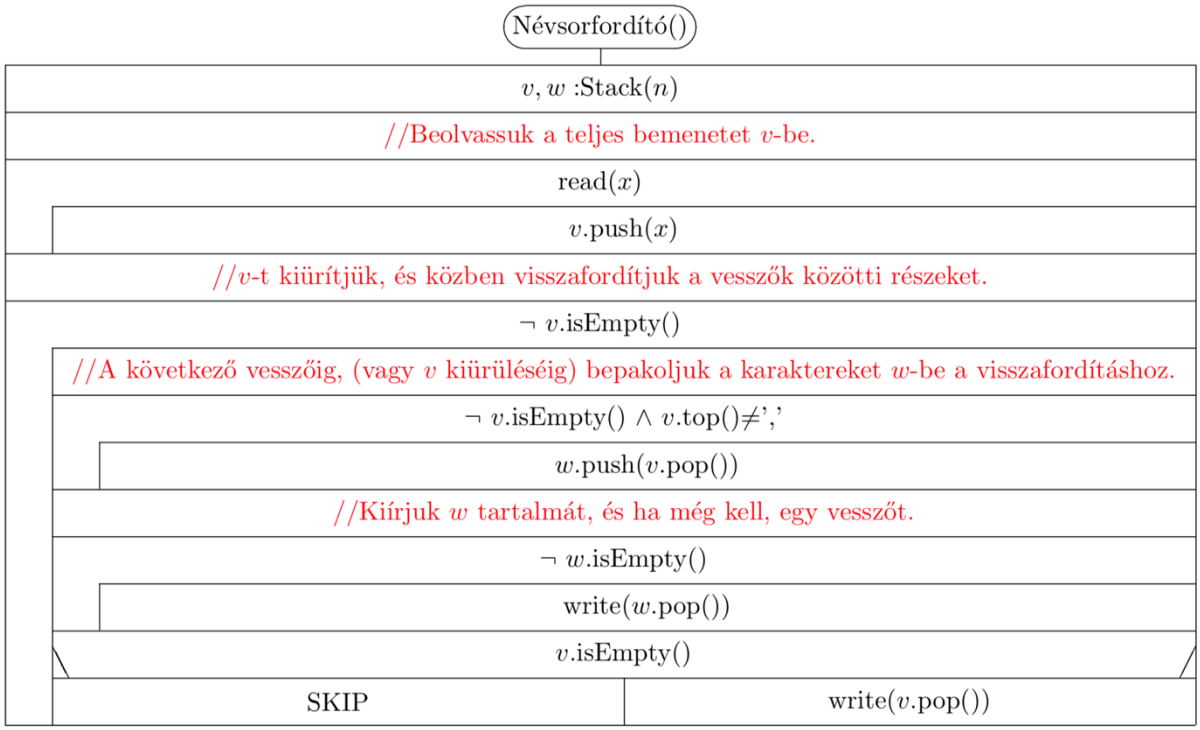
A beolvasáshoz az előadásjegyzetben is használt *read(&x: T) : 𝔹* függvényt használtuk, de ettől eltérhetünk. Ez a függvény beolvassa *x*-be a következő értéket a bemenetről, és ha ez sikeres, akkor igaz, ha pedig végére értünk az inputnak, akkor hamis lesz.

Vegyük észre, hogy a fenti módszerhez valójában nincs is szükség verem használatára. Elég lenne egyetlen számláló változóval nyilvántartani azt, hogy

eddig mennyivel több nyitó zárójellel találkoztunk, mint csukó zárójellel. Módosítsuk úgy a feladatot, hogy tényleg szükség legyen veremre: ezúttal írassuk ki az összetartozó zárójelpárok indexeit!



2. A bemenetről karakterenként beolvasunk egy (legfeljebb *n* karakterből álló) névsort, ahol a nevek egymástól vesszővel vannak elválasztva. Írjuk ki a neveket fordított sorrendben!
   Megoldás:



Javasolt házi feladat

Tükrözött-e a szöveg? Adott egy legfeljebb n hosszú, betűkből és '#' szimbólumokból álló sorozat. A sorozatot tükrözöttnek nevezzük, ha felbontható olyan páratlan hosszú, palindrom karaktersorozatokból álló részekre, amelyeknek középső karaktere az egyetlen bennük szereplő '#'. Döntsük el a bemenetről olvasott szövegről, hogy tükrözött-e?

Példa:

tükrözött: #, ####, abc#cba, ##a#aabc#cba

nem tükrözött: abc, abc#bc, abc#cbaa#aa, ab#bac##c