PPADS MMXXI

Matúš Jókay, C-503, matus.jokay@stuba.sk

uim.fei.stuba.sk/predmet/i-ppds

konzultácie elektronicky

mail, discord

LITTLE BOOK OF SEMAPHORES

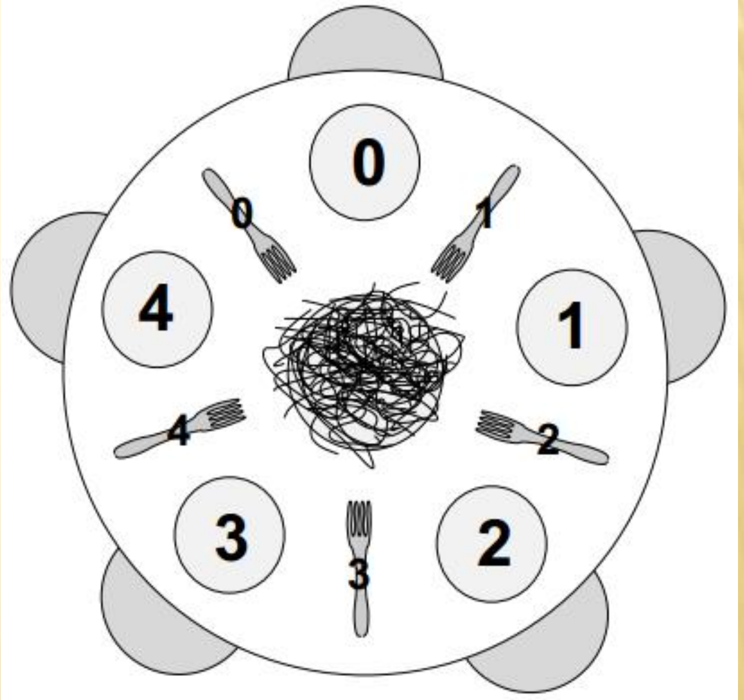
- Allen B. Downey: The Little Book of Semaphores, Version 2.2.1
- http://greenteapress.com/semaphores/LittleBookOfSemaphores.pdf

LITTLE BOOK OF SEMAPHORES

- Večerajúci filozofi (kapitola 4.4 Dining philosophers)

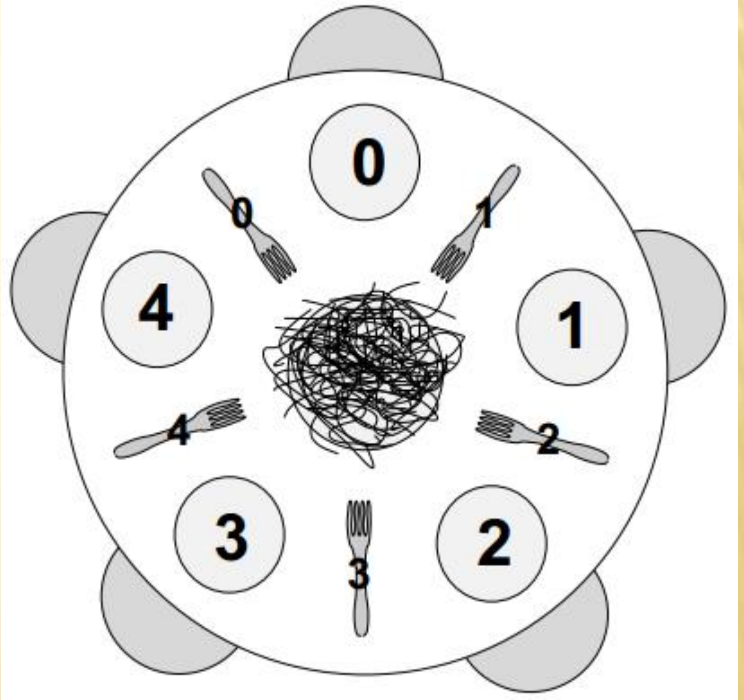
VEČERAJÚCI FILOZOFI

VEČERAJÚCI FILOZOFI



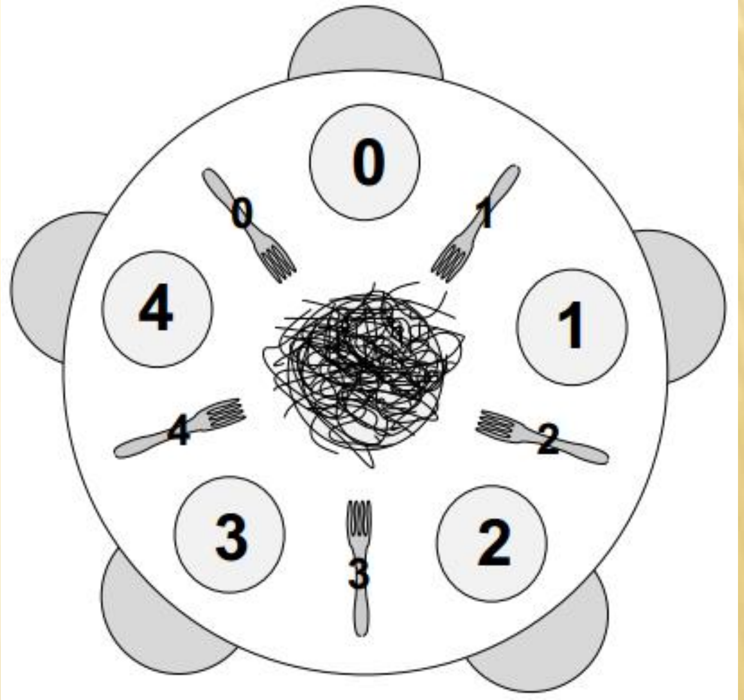
VEČERAJÚCI FILOZOFI

```
while True
```



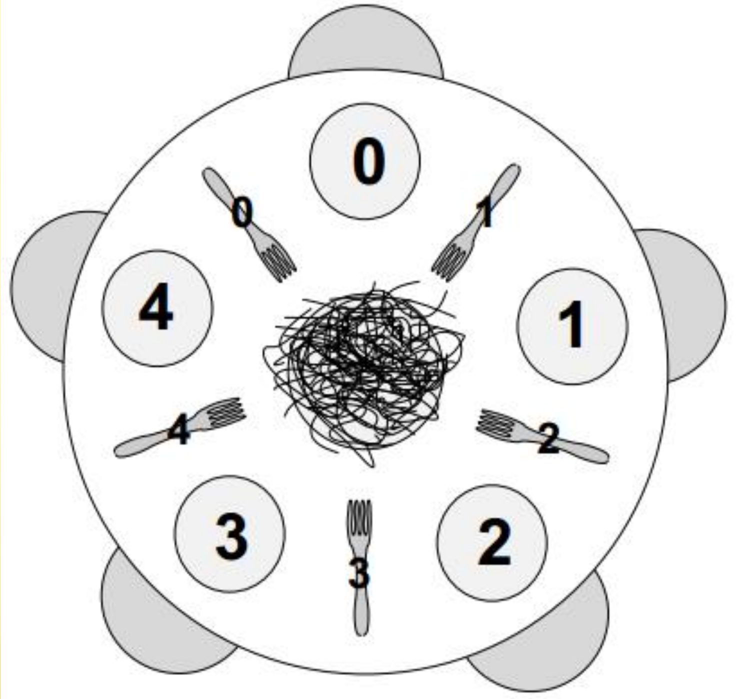
VEČERAJÚCI FILOZOFI

```
while True:
    think()
```



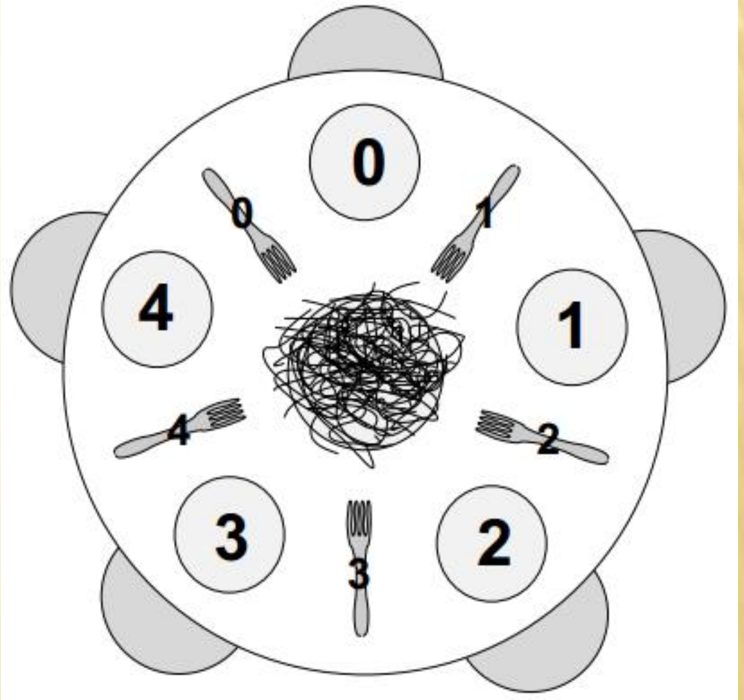
VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
```



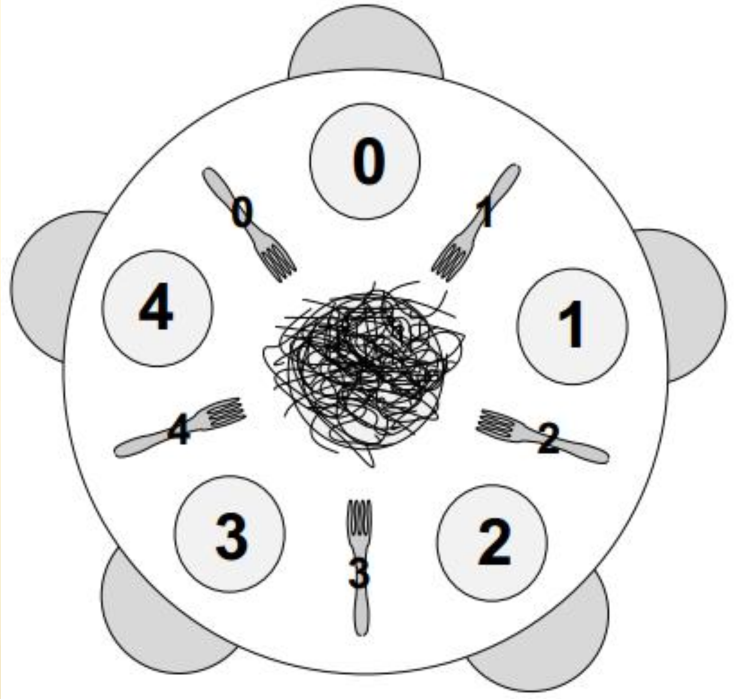
VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
```



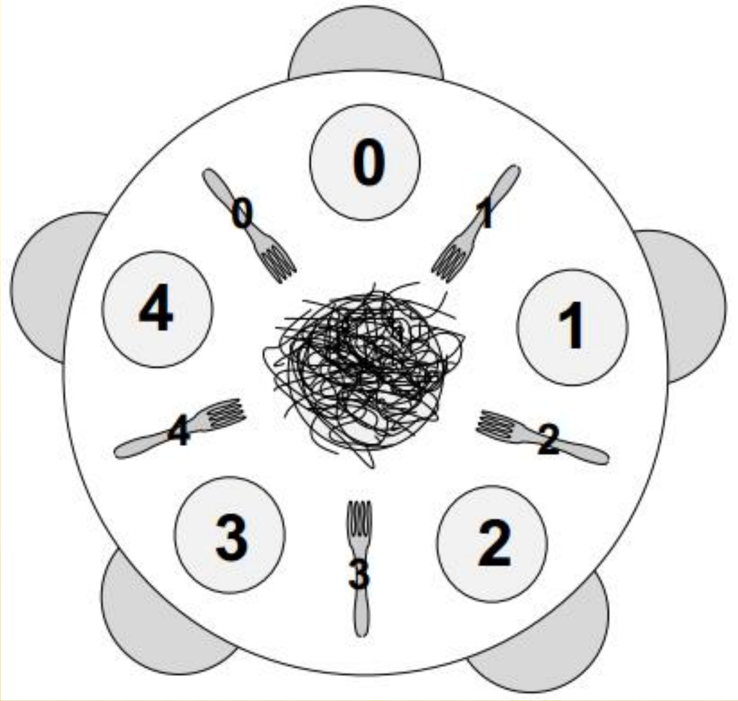
VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```



VEČERAJÚCI FILOZOFI

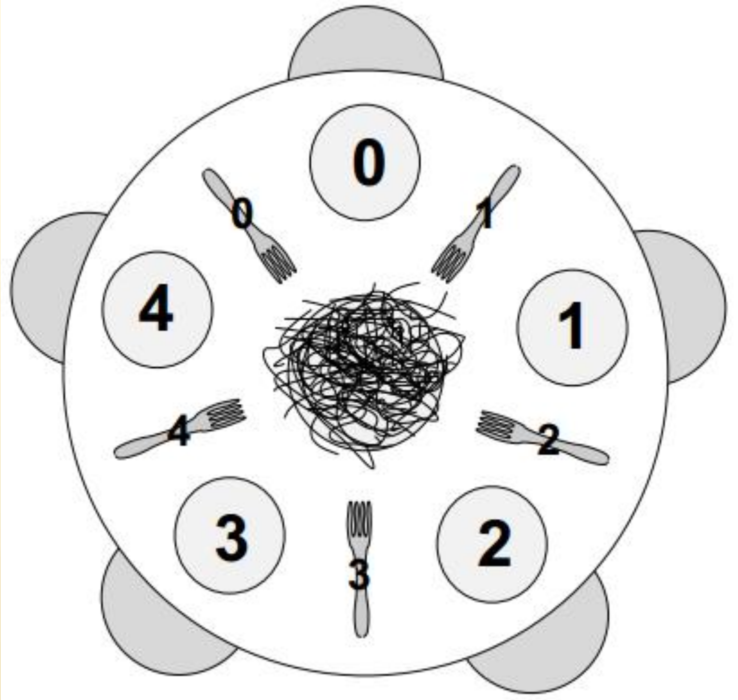
```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```



VEČERAJÚCI FILOZOFI

- Filozof → vlákno

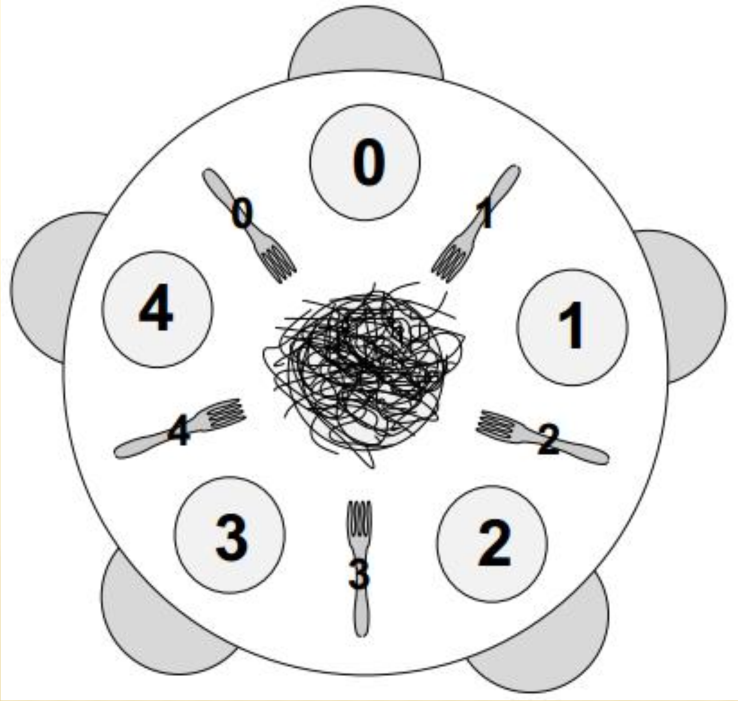
```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```



VEČERAJÚCI FILOZOFI

- Filozof → vlákno
- Vidlička → zdroj

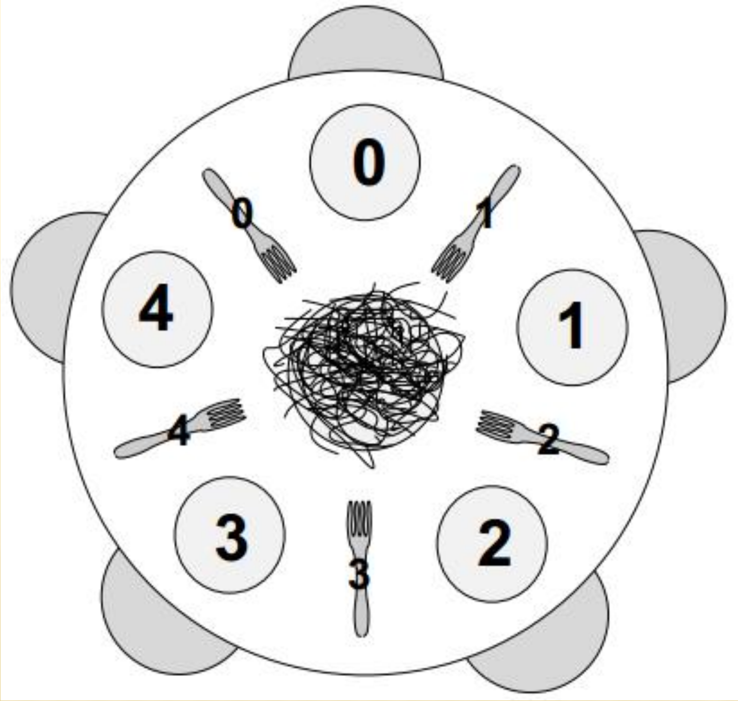
```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```



VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

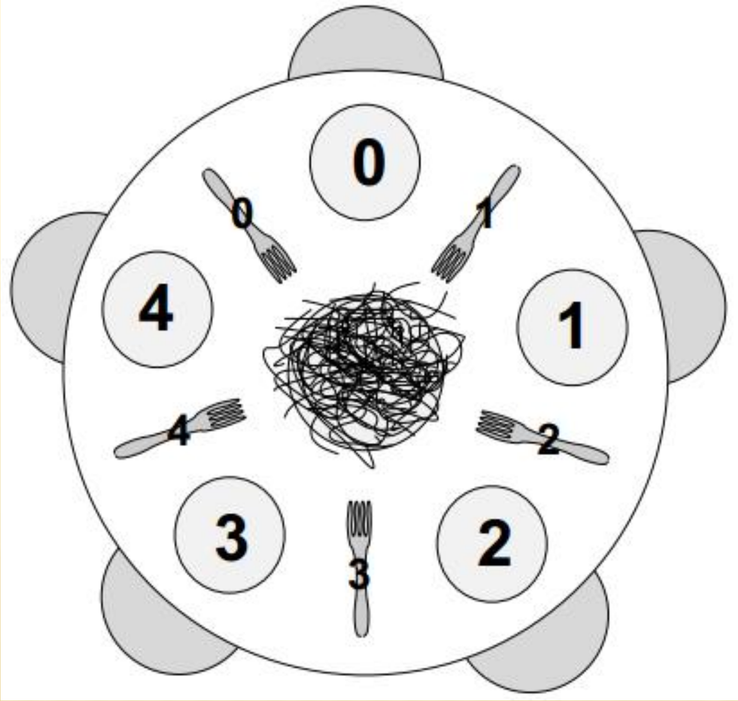
- Filozof → vlákno
- Vidlička → zdroj
- Čo je divné



VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

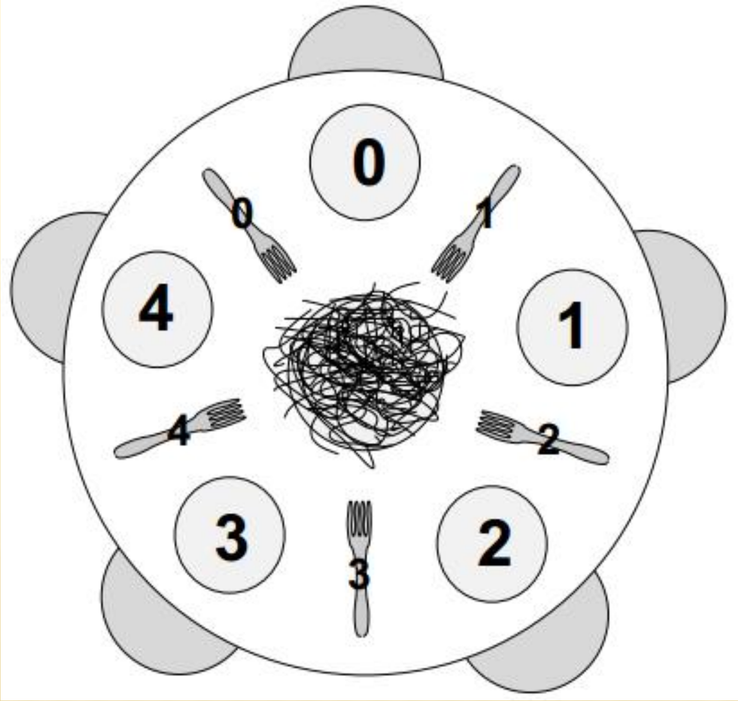
- Filozof → vlákno
- Vidlička → zdroj
- Čo je divné
  - Filozof potrebuje 2 vidličky



VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

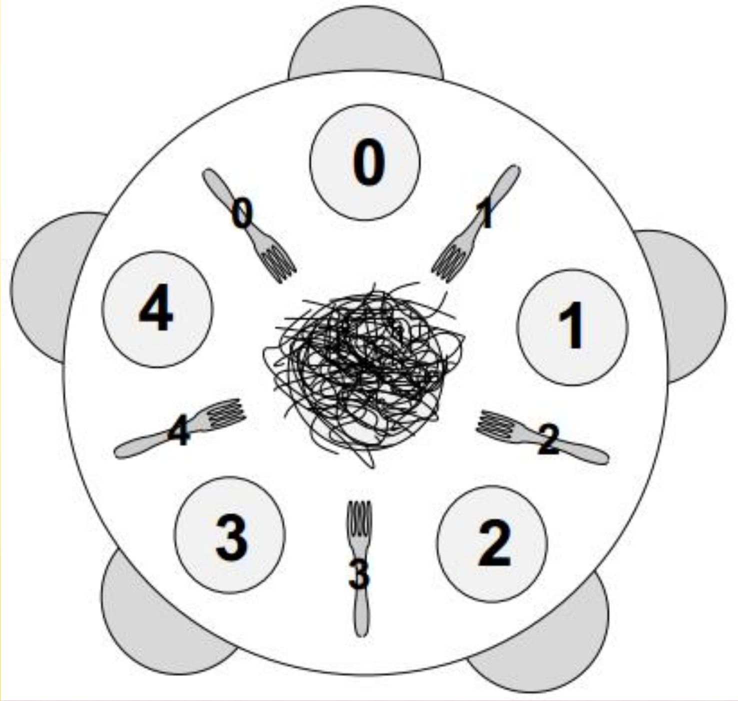
- Filozof → vlákno
- Vidlička → zdroj
- Čo je divné
  - Filozof potrebuje 2 vidličky
  - Ak má jednu, musí čakať na druhú, aby mohol pokračovat



VEČERAJÚCI FILOZOFI

- Filozof i

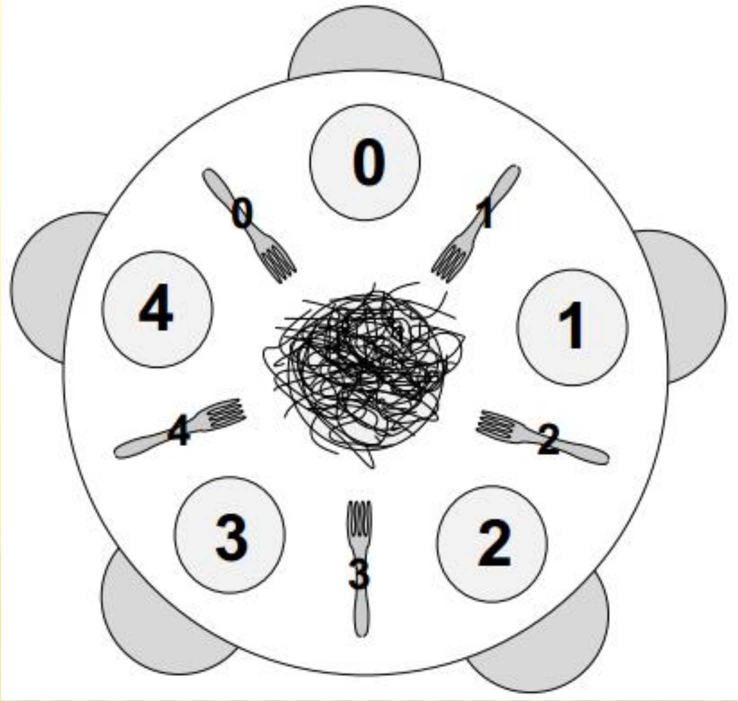
```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```



VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

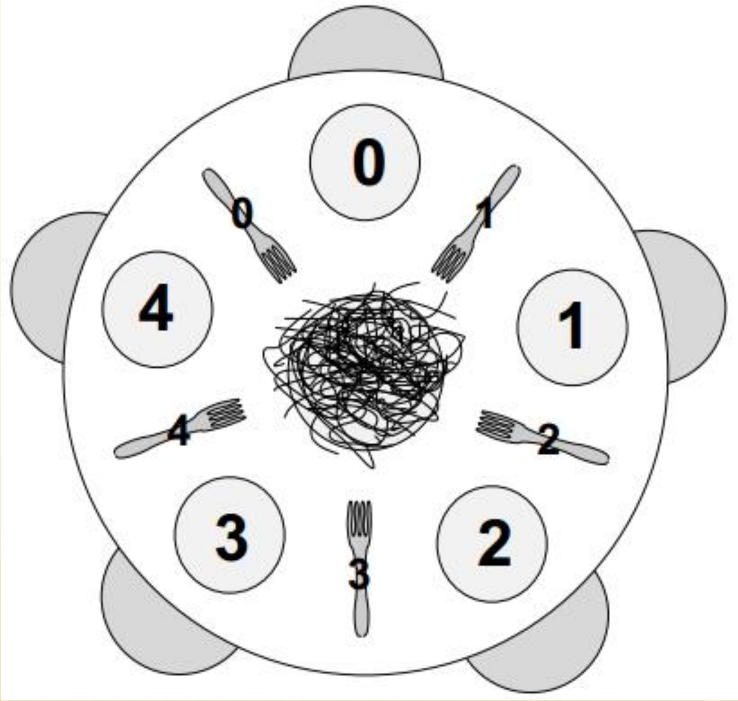
- Filozof i
  - Po pravej strane vidlička i



VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

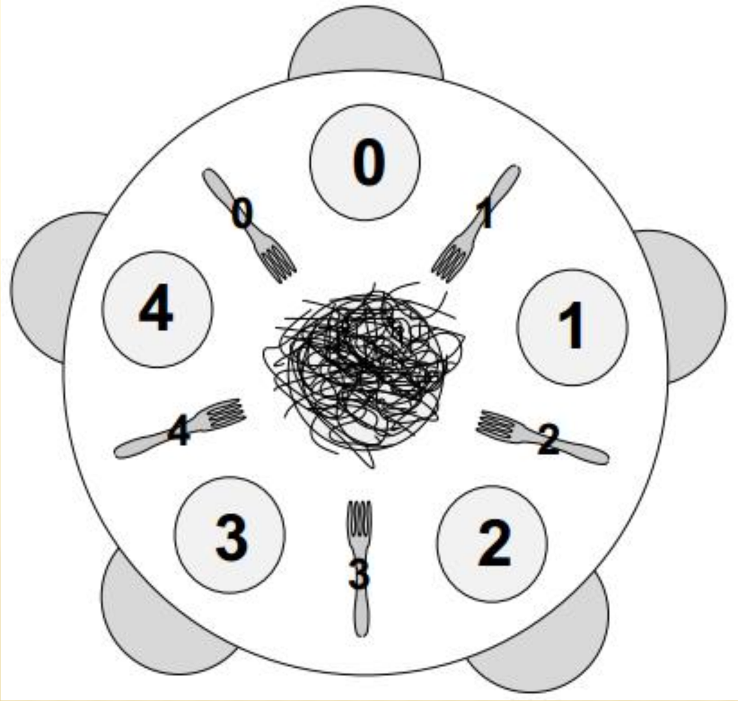
- Filozof i
  - Po pravej strane vidlička i
  - Po ľavej strane vidlička i+1



VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

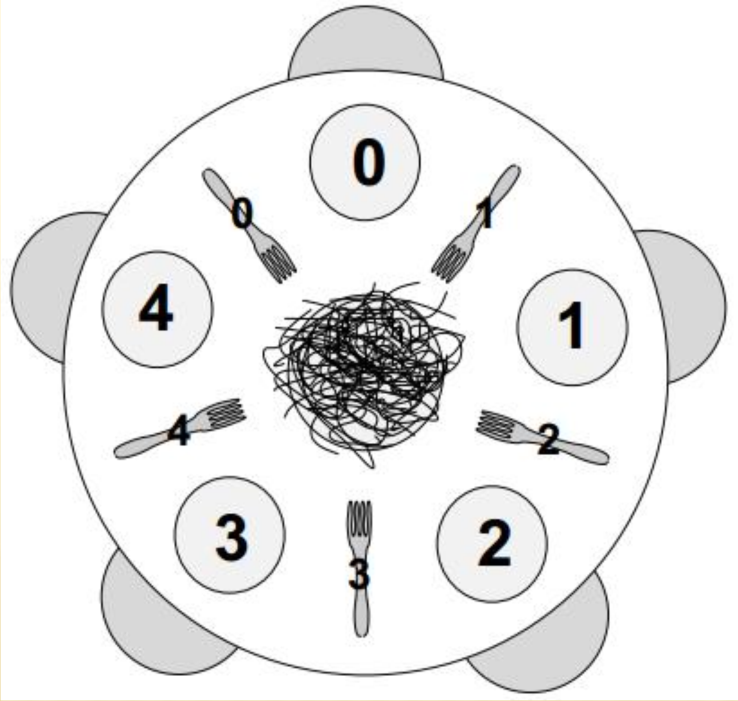
- Filozof i
  - Po pravej strane vidlička i
  - Po ľavej strane vidlička i+1
- i v intervale <0; 4>



```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

VEČERAJÚCI FILOZOFI

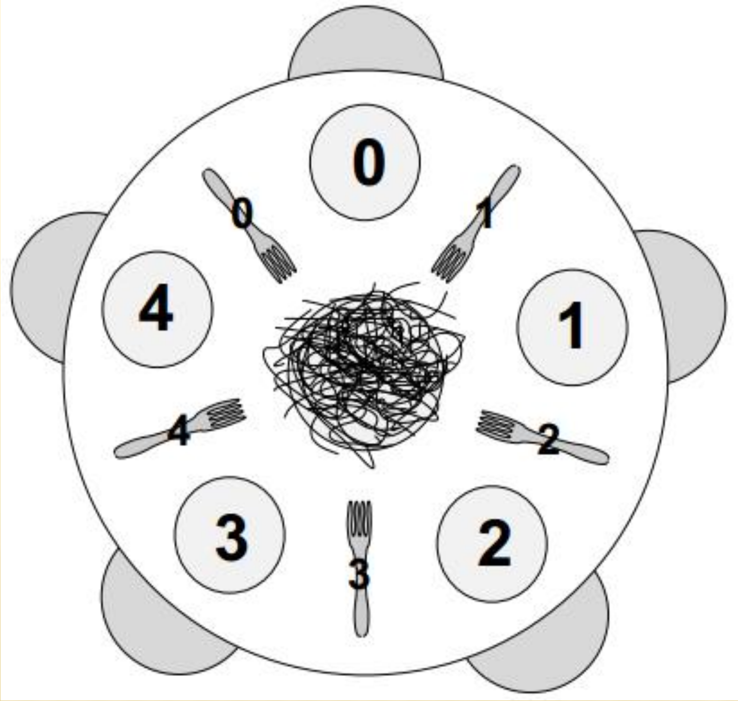
- Filozof i
  - Po pravej strane vidlička i
  - Po ľavej strane vidlička i+1
- i v intervale <0; 4>
- Filozof ovláda funkcie
  - think()
  - eat()



VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

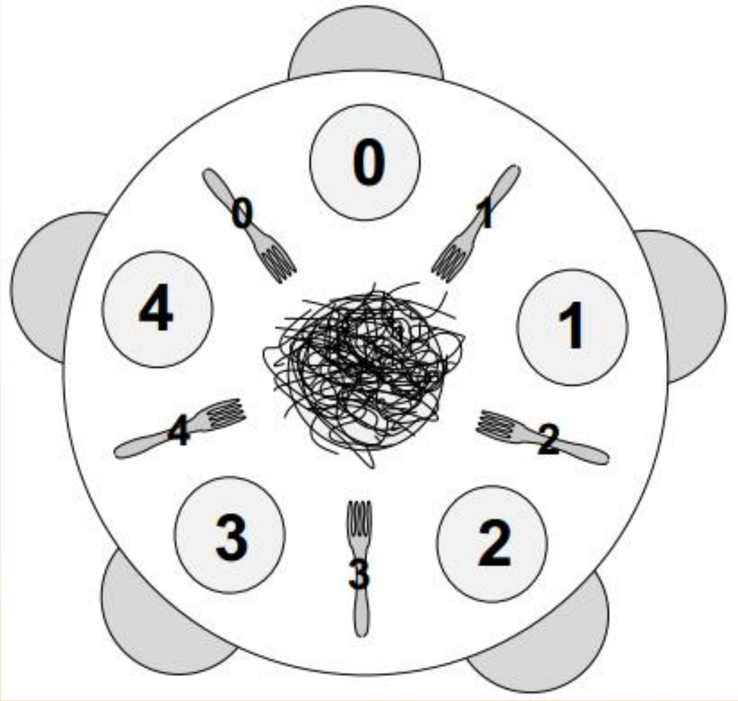
- Naša úloha definovať get() a put() tak, aby:



VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

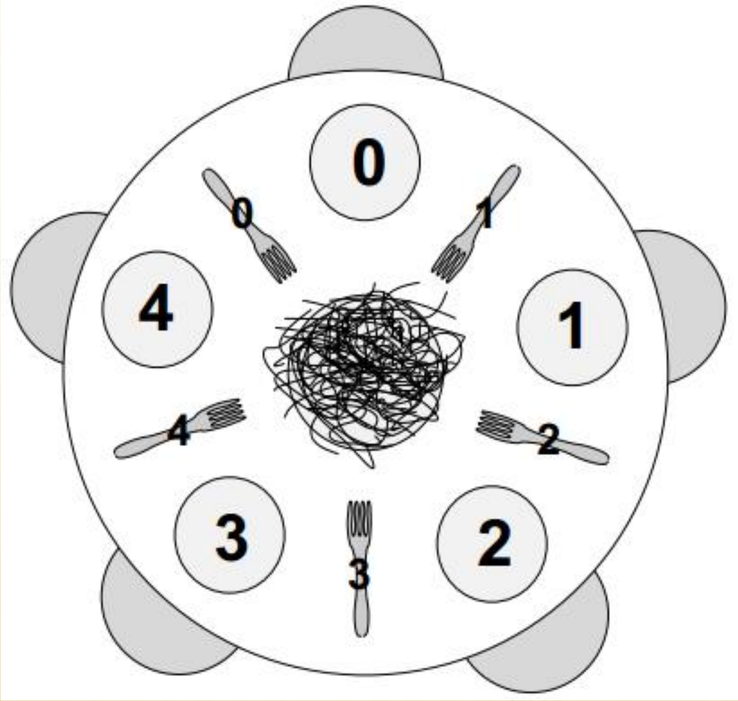
- Naša úloha definovať get() a put() tak, aby:
  1. V 1 čase držal vidličku iba jeden filozof



VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

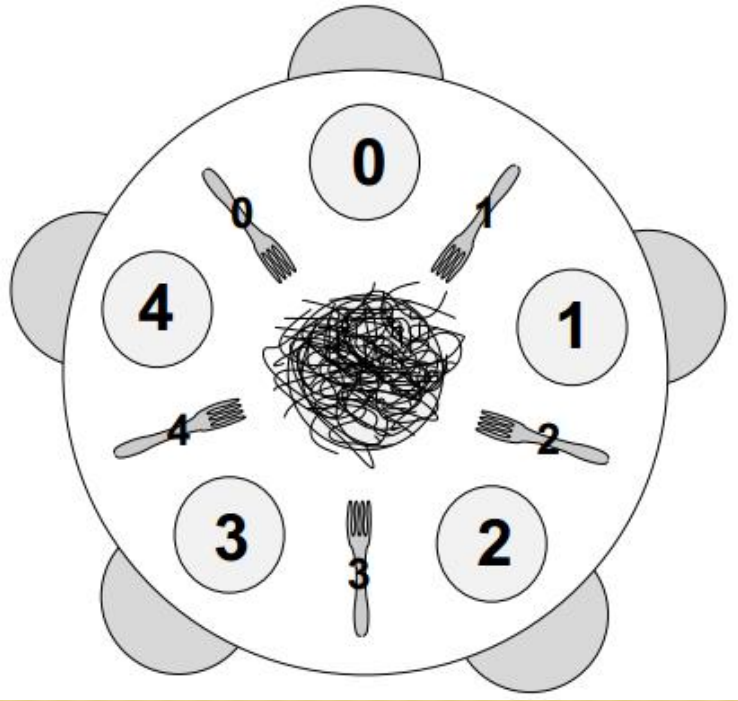
- Naša úloha definovať get() a put() tak, aby:
  1. V 1 čase držal vidličku iba jeden filozof
  2. Nenastalo uviaznutie



VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

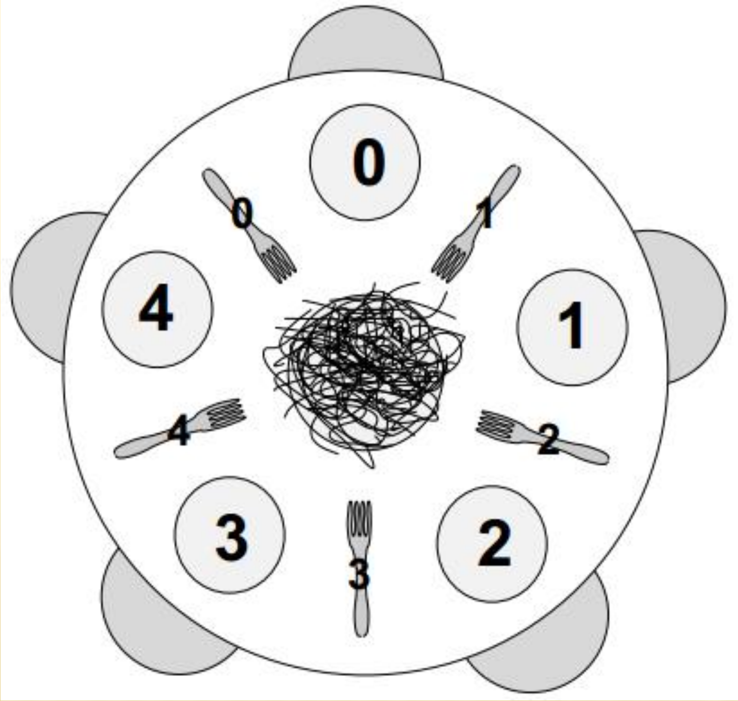
- Naša úloha definovať get() a put() tak, aby:
  1. V 1 čase držal vidličku iba jeden filozof
  2. Nenastalo uviaznutie
  3. Filozof neumrel od hladu



```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

VEČERAJÚCI FILOZOFI

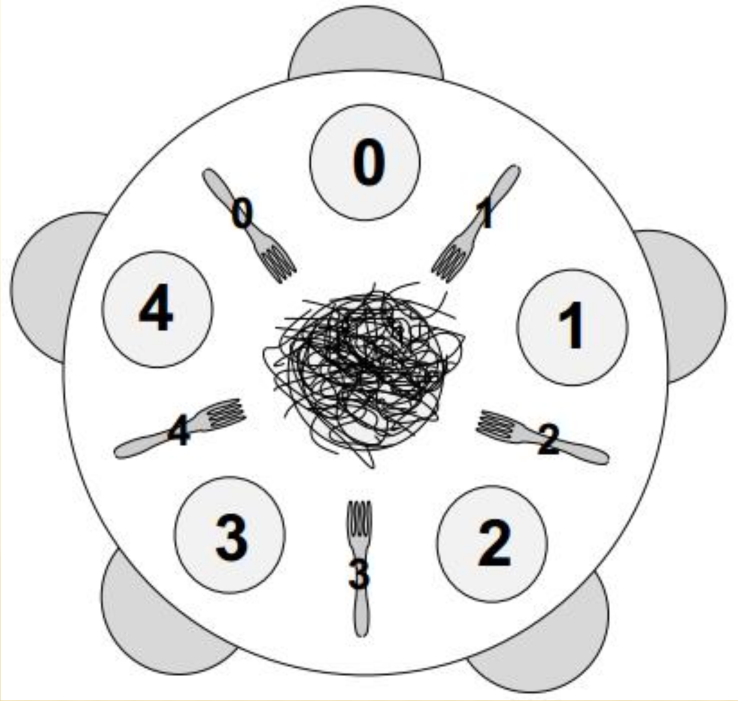
- Naša úloha definovať get() a put() tak, aby:
  1. V 1 čase držal vidličku iba jeden filozof
  2. Nenastalo uviaznutie
  3. Filozof neumrel od hladu
  4. Mohli jesť viacerí filozofi súčasne



VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

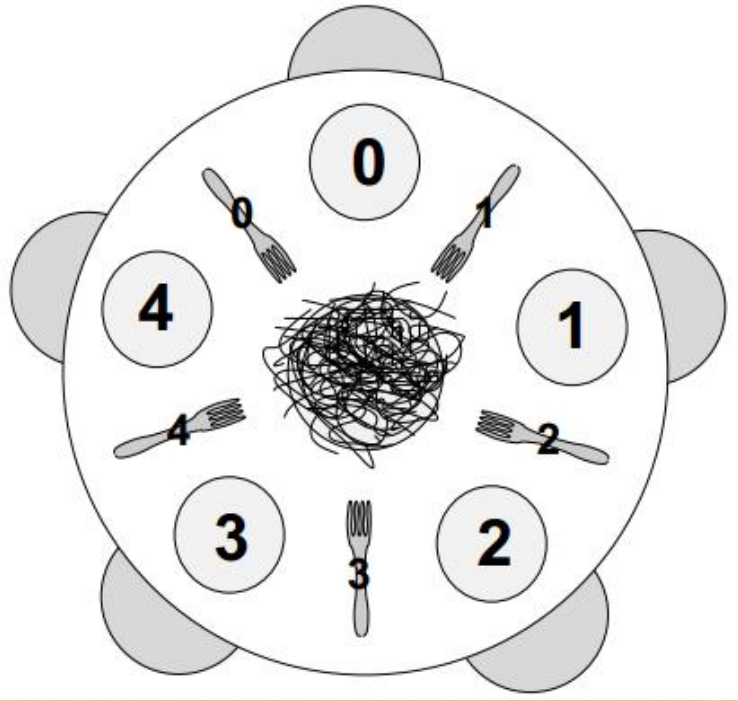
- Naša úloha definovať get() a put() tak, aby:
  1. V 1 čase držal vidličku iba jeden filozof
  2. Nenastalo uviaznutie
  3. Filozof neumrel od hladu
  4. Mohli jesť viacerí filozofi súčasne (konkurentne)



VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

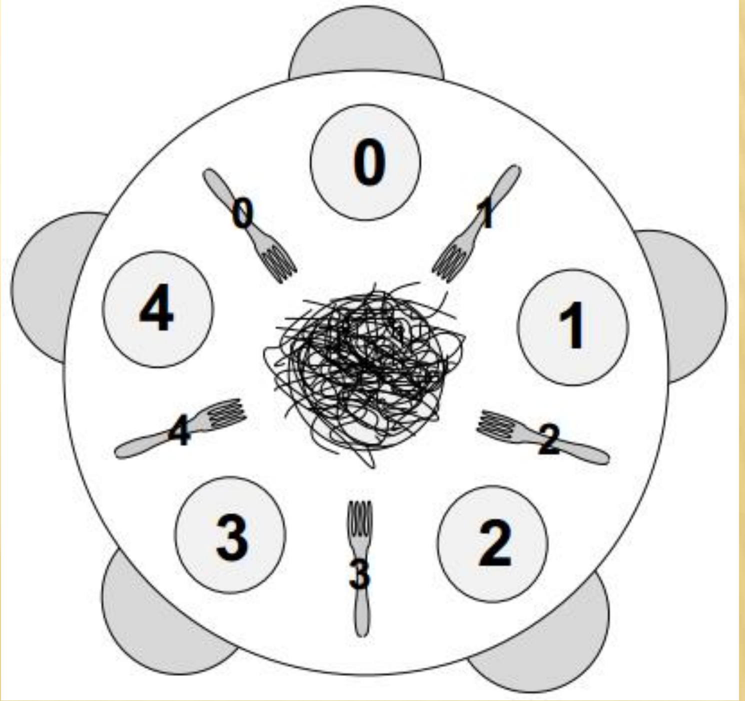
- Nepoznáme funkcie think() a eat()



VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

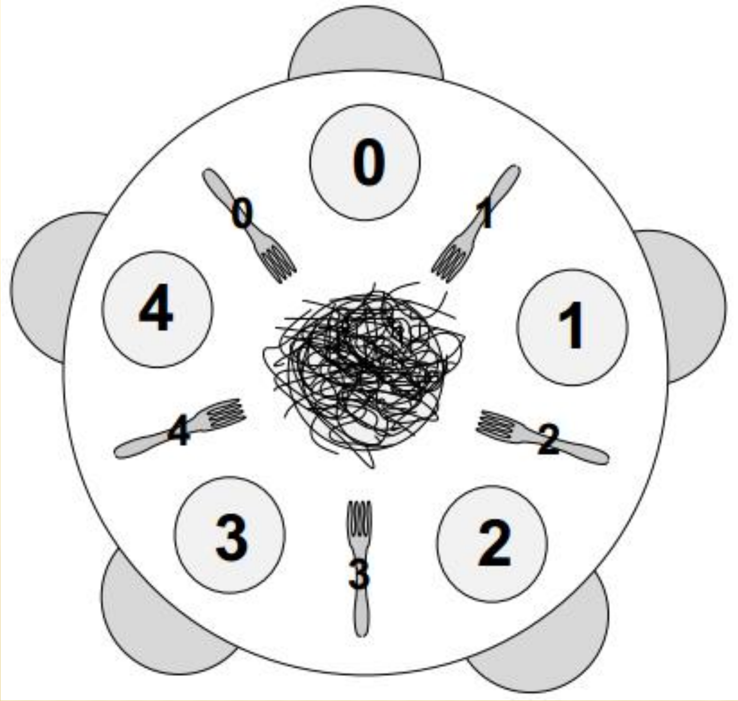
- Nepoznáme funkcie think() a eat(), ale:
  - think() nás z hľadiska synchronizácie nijako nezaujíma



VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

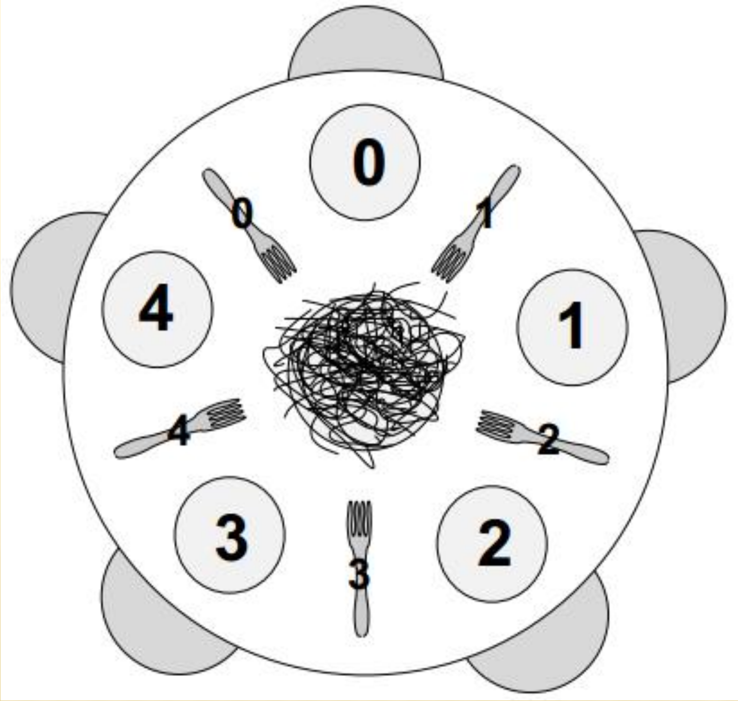
- Nepoznáme funkcie think() a eat(), ale:
  - think() nás z hľadiska synchronizácie nijako nezaujíma
  - eat() musí skončiť v konečnom čase



VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

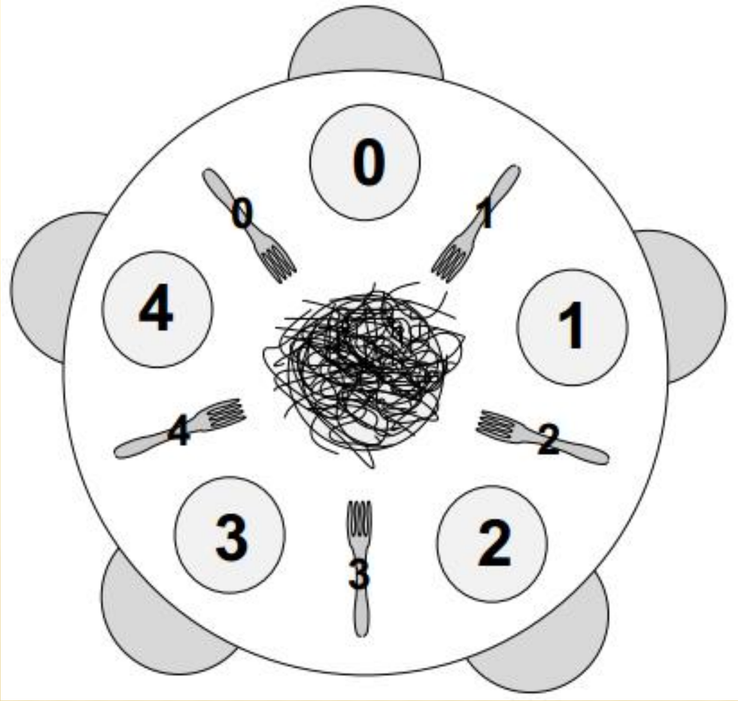
- Nepoznáme funkcie think() a eat(), ale:
    - think() nás z hľadiska synchronizácie nijako nezaujíma
    - eat() musí skončiť v konečnom čase, inak nesplníme podmienku 3!



VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

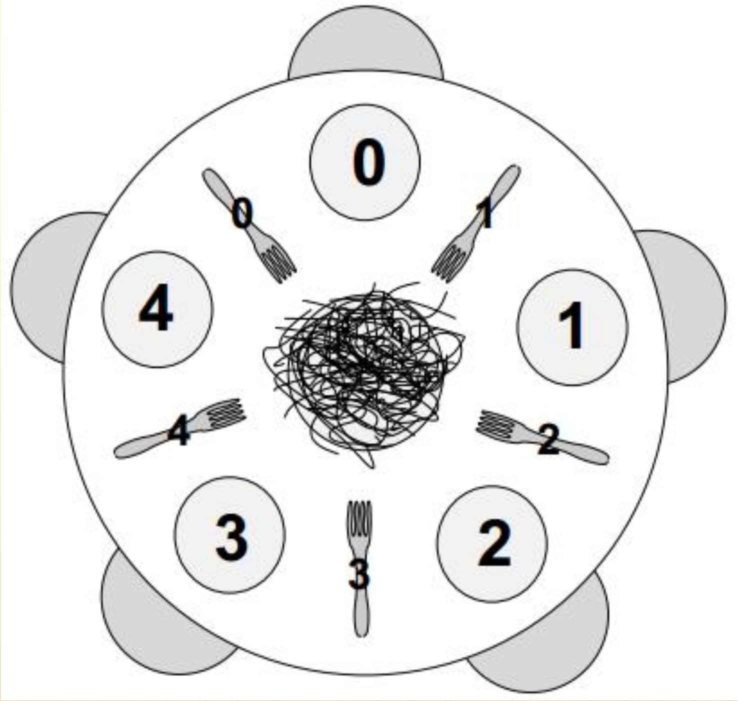
- Definujme pomocné funkcie na určenie id ľavej a pravej vidličky



```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

VEČERAJÚCI FILOZOFI

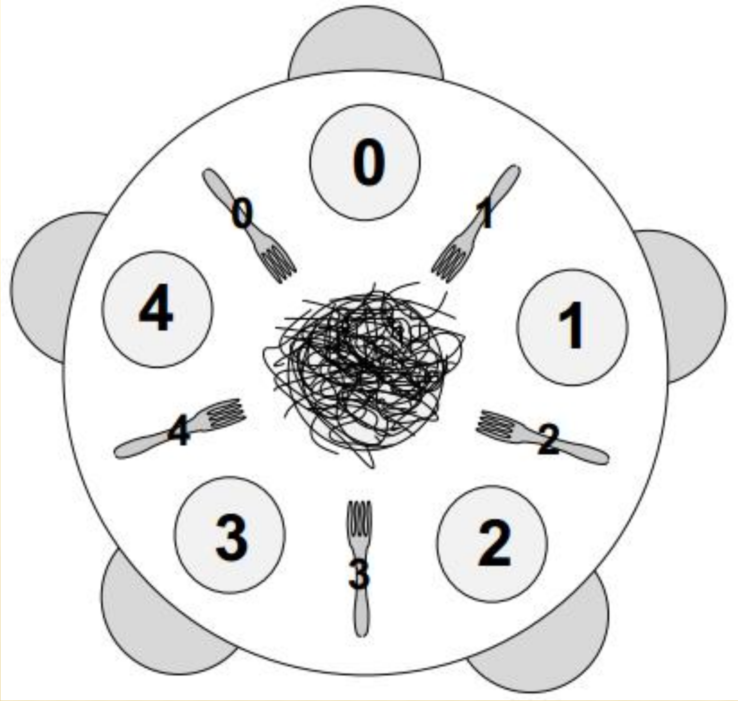
- Definujme pomocné funkcie na určenie id ľavej a pravej vidličky
- Keďže má pre vidličky platiť exkluzívny prístup



```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

VEČERAJÚCI FILOZOFI

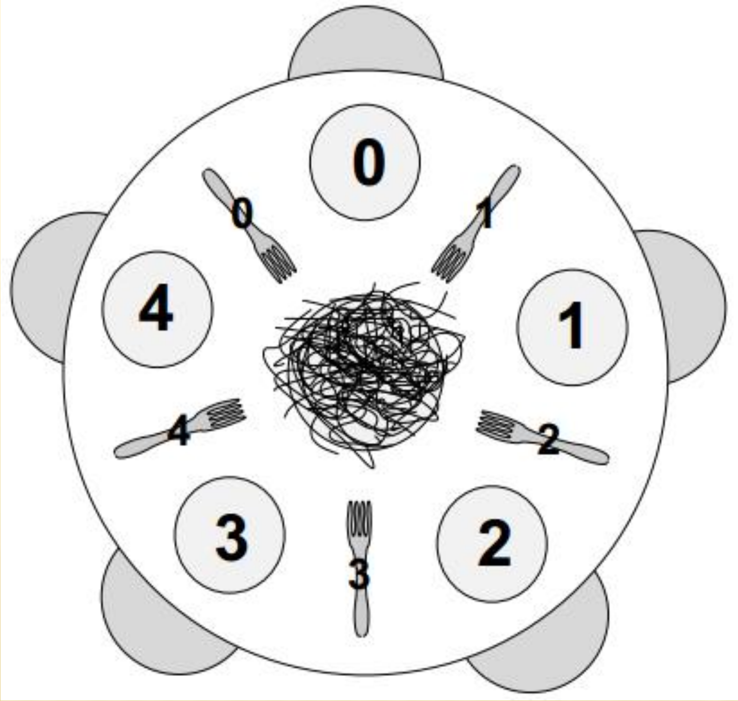
- Definujme pomocné funkcie na určenie id ľavej a pravej vidličky
- Keďže má pre vidličky platiť exkluzívny prístup, pre každú z nich definujme binárny semafor



VEČERAJÚCI FILOZOFI

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```



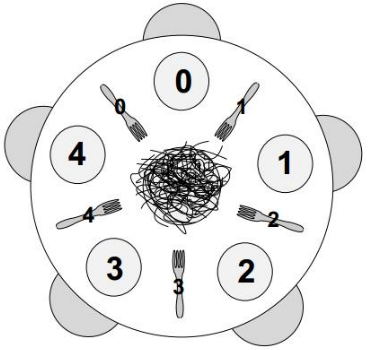
VEČERAJÚCI FILOZOFI #1

```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```



VEČERAJÚCI FILOZOFI #1

```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

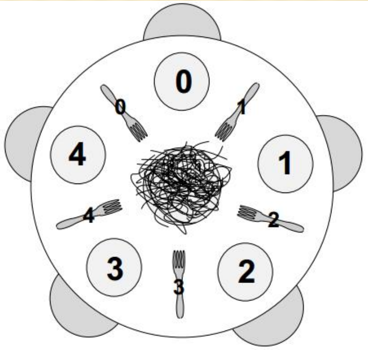
funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

funkcia put_forks

```
def put_forks(i):
    forks[right(i)].signal()
    forks[left(i)].signal()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```



VEČERAJÚCI FILOZOFI #1

```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

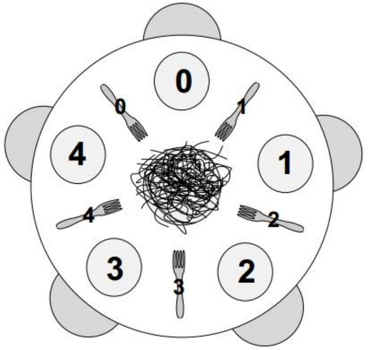
```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

funkcia put_forks

```
def put_forks(i):
    forks[right(i)].signal()
    forks[left(i)].signal()
```

?

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```



VEČERAJÚCI FILOZOFI #1

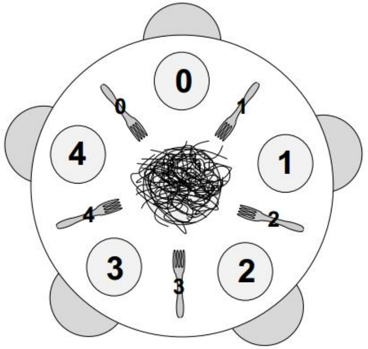
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Spĺňame podmienku 1?



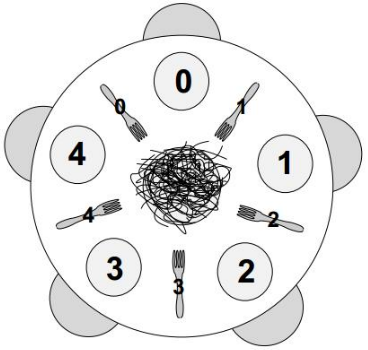
VEČERAJÚCI FILOZOFI #1

```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

```
funkcia get_forks
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Spĺňame podmienku 1? (v 1 čase drží vidličku 1 filozof)



VEČERAJÚCI FILOZOFI #1

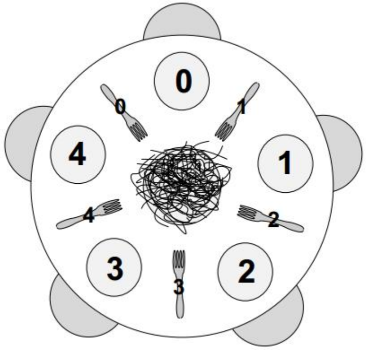
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Spĺňame podmienku 1? (v 1 čase drží vidličku 1 filozof)
- Spĺňame podmienku 2?



VEČERAJÚCI FILOZOFI #1

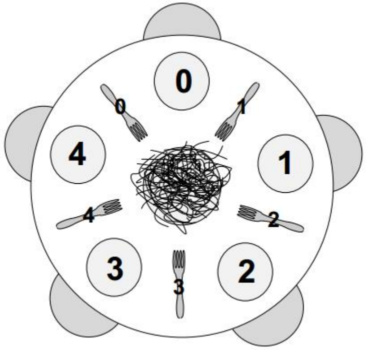
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Spĺňame podmienku 1? (v 1 čase drží vidličku 1 filozof)
- Spĺňame podmienku 2? (nenastane uviaznutie)



VEČERAJÚCI FILOZOFI #1

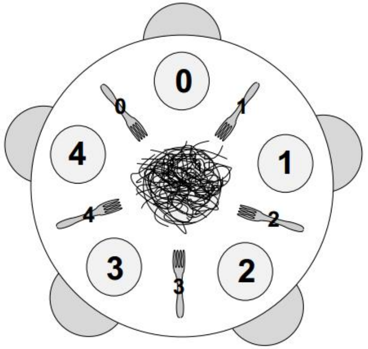
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Prečo nastáva uviaznutie?



VEČERAJÚCI FILOZOFI #1

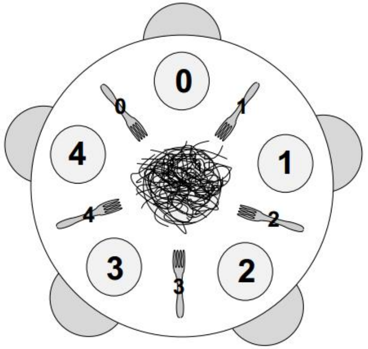
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Prečo nastáva uviaznutie?
  1. Okrúhly stol



VEČERAJÚCI FILOZOFI #1

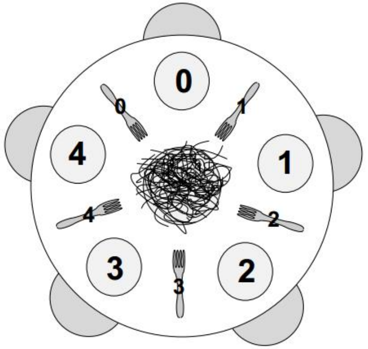
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

- Prečo nastáva uviaznutie?
  1. Okrúhly stol
  2. Všetci môžu súčasne jesť

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```



VEČERAJÚCI FILOZOFI #1

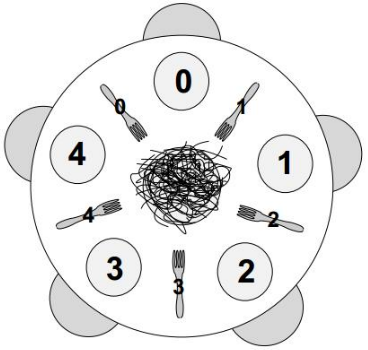
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Prečo nastáva uviaznutie?
  1. Okrúhly stol
  2. Všetci môžu súčasne jesť
  3. Všetci berú najprv pravú vidličku



VEČERAJÚCI FILOZOFI #1

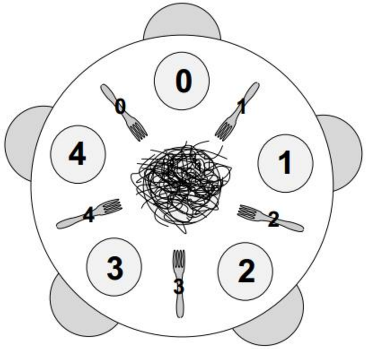
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Ako odstránime uviaznutie?



VEČERAJÚCI FILOZOFI #1

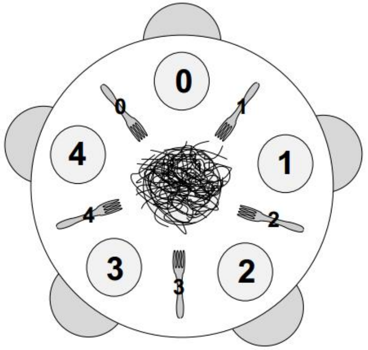
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Ako odstránime uviaznutie?
- Zrušíme aspoň **jednu** z podmienok uviaznutia



VEČERAJÚCI FILOZOFI #1

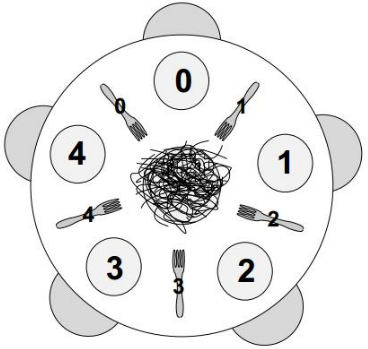
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

- Ako odstránime uviaznutie?
  - Zmeníme stôl?

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```



VEČERAJÚCI FILOZOFI #1

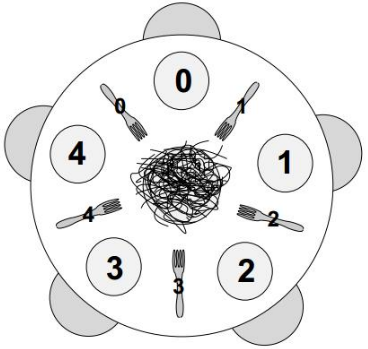
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Ako odstránime uviaznutie?
  - Zmeníme stôl? Asi nie…



VEČERAJÚCI FILOZOFI #1

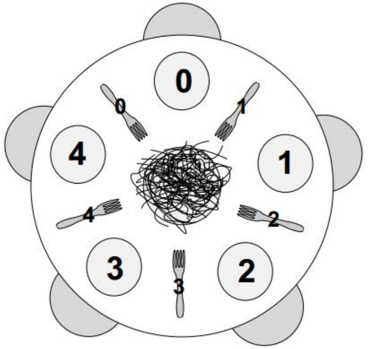
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Ako odstránime uviaznutie?
  - Zmeníme stôl? Asi nie…
  - Umožníme jesť naraz max 4?



VEČERAJÚCI FILOZOFI #1

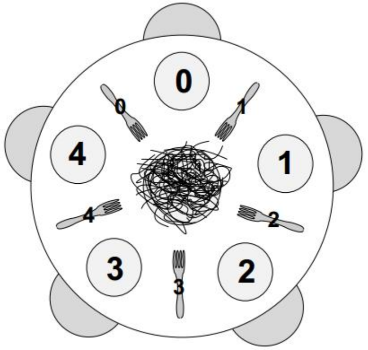
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Ako odstránime uviaznutie?
    - Zmeníme stôl? Asi nie…
    - Umožníme jesť naraz max 4? Ako?



VEČERAJÚCI FILOZOFI #1

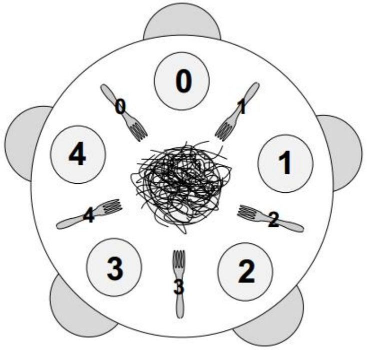
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Ako odstránime uviaznutie?
  - Zmeníme stôl? Asi nie…
  - Umožníme jesť naraz max 4? Ako?
  - Zavedieme ľavákov a pravákov?



VEČERAJÚCI FILOZOFI #1

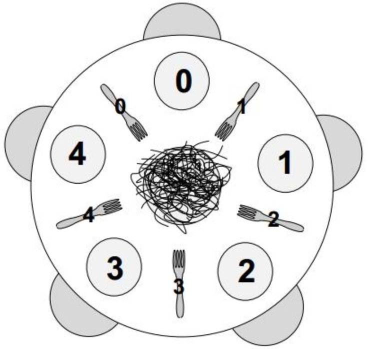
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Ako odstránime uviaznutie?
  - Zmeníme stôl? Asi nie…
  - Umožníme jesť naraz max 4? Ako?
  - Zavedieme ľavákov a pravákov? Ako?



VEČERAJÚCI FILOZOFI #2

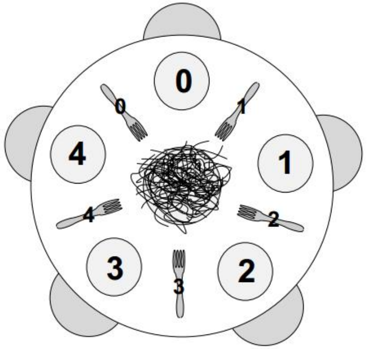
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

- Skúsme riešiť…

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```



VEČERAJÚCI FILOZOFI #2

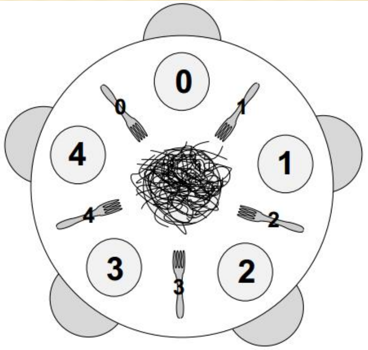
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Skúsme riešiť… obmedzenie max počtu tých, ktorí môžu naraz jesť



VEČERAJÚCI FILOZOFI #2

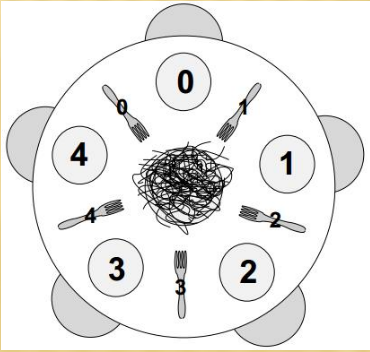
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Zavedieme čašníka (footman)



VEČERAJÚCI FILOZOFI #2

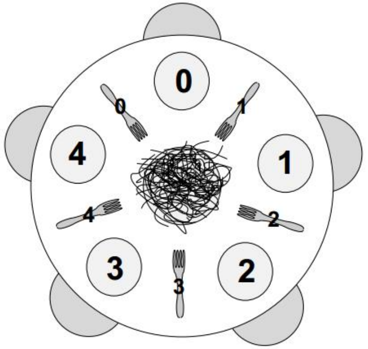
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Zavedieme čašníka (footman)
  - semafor



VEČERAJÚCI FILOZOFI #2

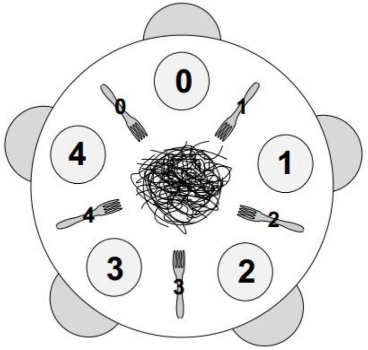
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Zavedieme čašníka (footman)
  - semafor
  - inicializovaný na max počet naraz jeduvších



VEČERAJÚCI FILOZOFI #2

```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

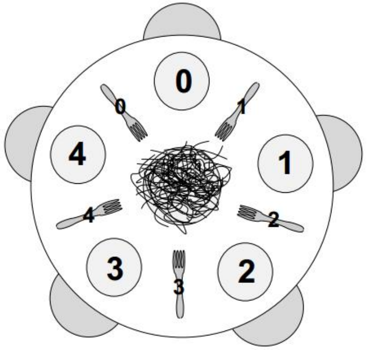
funkcia get_forks

```
def get_forks(i):

    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

funkcia put_forks

```
def put_forks(i):
    forks[right(i)].signal()
    forks[left(i)].signal()

```



VEČERAJÚCI FILOZOFI #2

```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

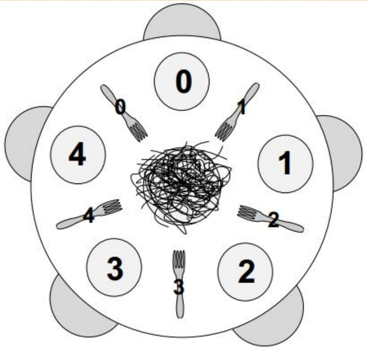
funkcia get_forks

```
def get_forks(i):
    footman.wait()
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

funkcia put_forks

```
def put_forks(i):
    forks[right(i)].signal()
    forks[left(i)].signal()
    footman.signal()
```



VEČERAJÚCI FILOZOFI #2

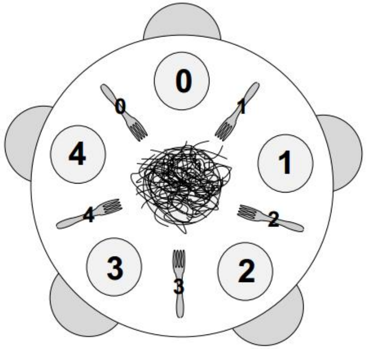
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    footman.wait()
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Podmienka 1 (1 v pre 1 f v 1 c)



VEČERAJÚCI FILOZOFI #2

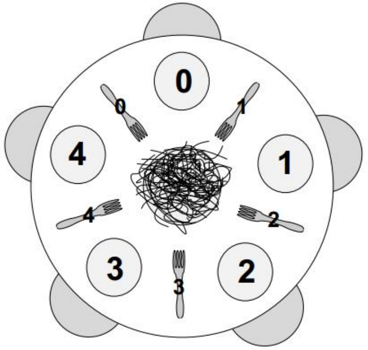
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    footman.wait()
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Podmienka 1 (1 v pre 1 f v 1 c)
- Podmienka 2 (deadlock)



VEČERAJÚCI FILOZOFI #2

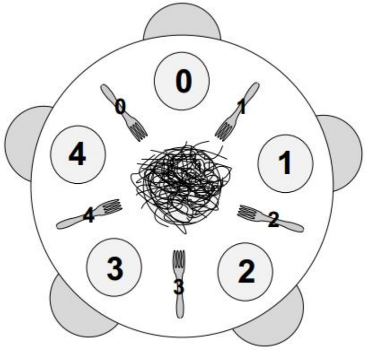
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    footman.wait()
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Podmienka 1 (1 v pre 1 f v 1 c)
- Podmienka 2 (deadlock)
- Podmienka 3 (vyhladnutie)



VEČERAJÚCI FILOZOFI #2

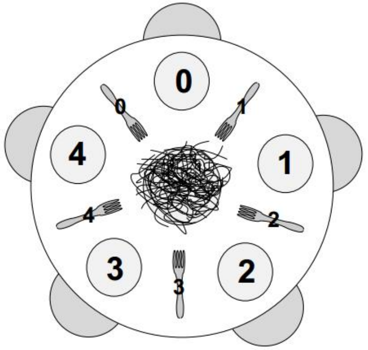
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    footman.wait()
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Podmienka 1 (1 v pre 1 f v 1 c)
- Podmienka 2 (deadlock)
- Podmienka 3 (vyhladnutie)
- Podmienka 4 (konkurentnosť)



VEČERAJÚCI FILOZOFI #3

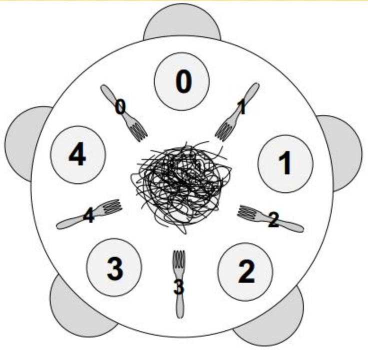
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Ako odstránime uviaznutie?
  - Zmeníme stôl? NIE
  - Umožníme jesť naraz max 4? ÁNO
  - Zavedieme ľavákov a pravákov? Ako?



VEČERAJÚCI FILOZOFI #3

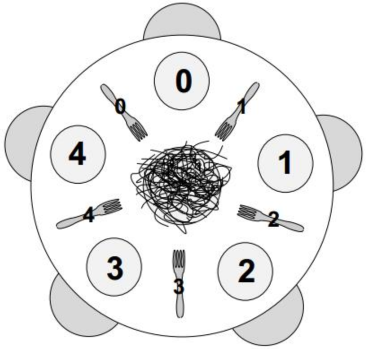
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

- Ľaváci a praváci

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```



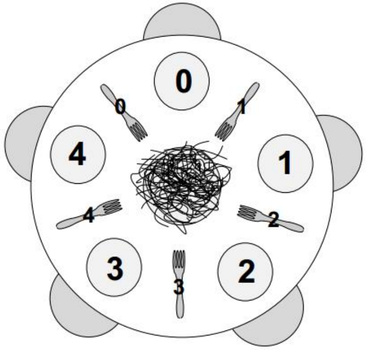
VEČERAJÚCI FILOZOFI #3

```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

```
funkcia get_forks
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Ľaváci a praváci
  - Aspoň 1 ľavák, aspoň 1 pravák



VEČERAJÚCI FILOZOFI #3

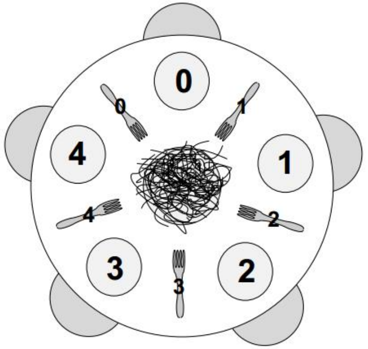
```
def right: return i
def left: return (i+1) % 5

forks = [Semaphore(1) for i in range(5)]
```

funkcia get_forks

```
def get_forks(i):
    forks[right(i)].wait()
    forks[left(i)].wait()
```

```
while True:
    think()
    get_forks()
    eat()
    put_forks()
```

- Ľaváci a praváci
  - Aspoň 1 ľavák, aspoň 1 pravák
  - Ako to funguje?



CVIČENIE

- Prvá časť: obe riešenia problému filozofov
- Druhá časť: analýza a riešenie zápočtového príkladu z roku 2017