

---

Algoritmai ir programavimas

Rekursija

---

---

# Tikslai

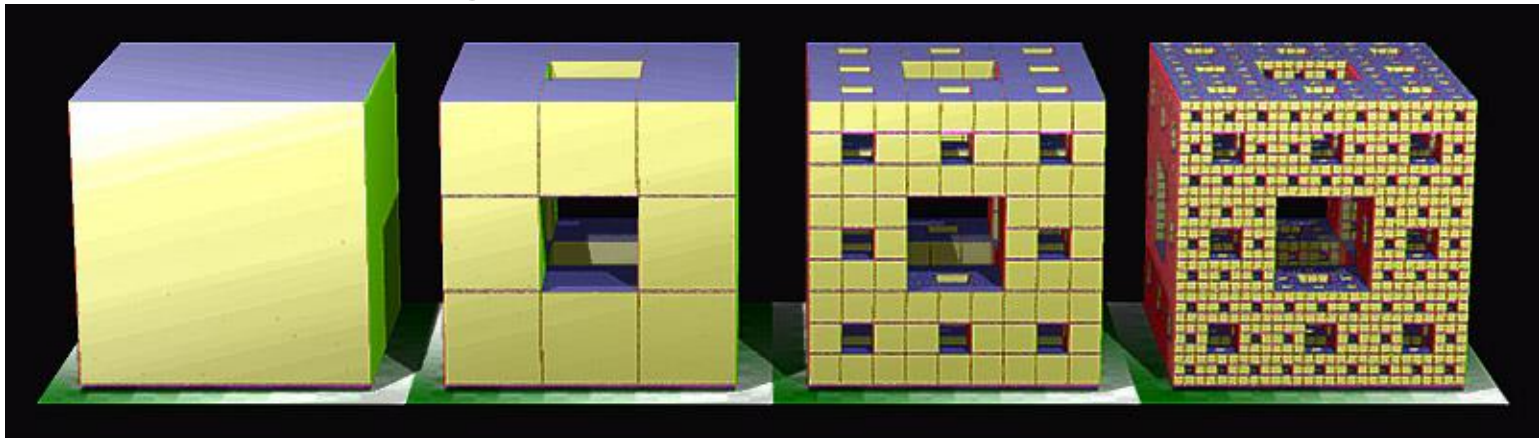
- Suprasti rekursijos sąvoką ir principus
- Išmokti rašyti rekursines funkcijas programavimo kalboje
- Suvokti rekursijos taikymo sritis ir privalumus/trūkumus

# Kas yra rekursija

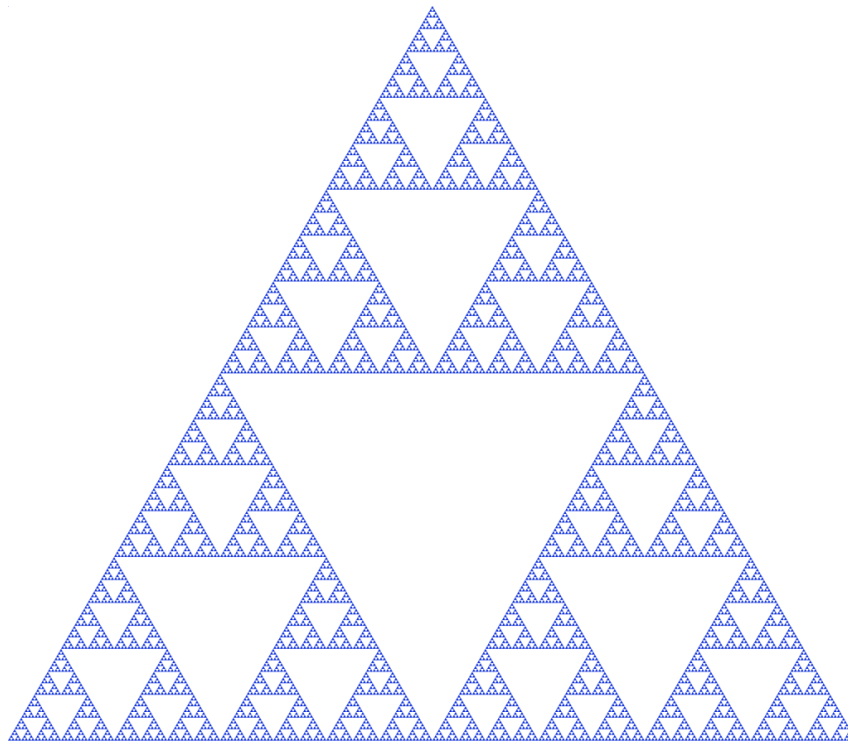
*Rekursija – tai procesas, kai funkcija kviečia pati save spręsti mažesnę problemos versiją*

# Mengerio kemppane (geometrinė figūra)

- Konstravimo algoritmas:
  - Paimamas kubas.
  - Kubas suskaidomas į 27 vienodo dydžio kubelius.
  - Pašalinamas kubo viduryje esantis kubelis, taip pat dar 6 kiekvienos sienos viduryje esantys kubeliai.
  - Toliau su kiekvienu likusiu kubeliu veiksmai kartojami nuo antro žingsnio



# Sierpinskio trikampis



\* Šaltinis: [https://en.wikipedia.org/wiki/Sierpi%C5%84ski\\_triangle](https://en.wikipedia.org/wiki/Sierpi%C5%84ski_triangle)

# Pavyzdys iš realaus gyvenimo

„Veidrodžių kambarys“  
– stovint tarp dviejų  
veidrodžių matome  
begalinį vaizdo  
atspindį, panašiai  
veikia ir rekursija



# Rekursijos pavyzdys (Faktorialo skaičiavimas)

```
def faktorialas(n):  
    if n == 0:  
        return 1  
    else:  
        return n * faktorialas(n-1)
```

Python

```
long long fakt(int n) {  
    if(n == 0) return 1;  
    return n*fakt(n-1);  
}
```

C++

# Rekursijos terminai

- **Bazinis atvejis:** momentas, kai rekursija sustoja (pvz., kai  $n == 0$ ).
- **Rekursinis atvejis:** funkcija kviečia pati save sprendama mažesnę problemos versiją



# Praktika

- Apskaičiuoti Fibonačio seką

***Fibonacci seka (Fibonāčio sekà), skaičių seka, kurios kiekvienas paskesnis narys lygus dviejų prieš jį esančių narių sumai, t. y.  $F_{i+2} = F_{i+1} + F_i$ ,  $i = 1, 2, \dots$ ;  $F_1 = F_2 = 1$ . Pirmieji sekos nariai yra 1, 1, 2, 3, 5, 8, 13, 21, ....***

1, 1, 2, 3, 5, 8, 13, 21, 34...

# Fibonačio seka

```
def fibonacci(n):  
    if n <= 1:  
        return n  
    else:  
        return fibonacci(n-1) + fibonacci(n-2)
```

Python

```
long long F(int n) {  
    if(n <= 2) return 1;  
    return F(n-1) + F(n-2);  
}
```

C++

# Kur naudojama rekursija?

- Algoritmai: binarės paieškos medis, rūšiavimas, tokie kaip greitojo rūšiavimo algoritmas.
- Matematinės problemos: skaičių sekos, kombinatorika.
- Žaidimų kūrimas: algoritmai, sprendžiantys galvosūkius, maršrutų paieška

---

# Rekursijos privalumai

- Lengvesnis problemos supratimas ir sprendimas
- Švaresnis ir elegantiškesnis kodas
- Natūralus medžių struktūrų sprendimas
- Tinka tam tikrų algoritmų įgyvendinimui
- Intuityvus sprendimų modeliavimas
- Naudojimas, kai tiksli iteracija nežinoma

# Rekursijos trūkumai

- Didelis atminties naudojimas
- Lėtesnis veikimas
- Sunkesnis supratimas ir derinimas
- Rizika įstrigti begalinėje rekursijoje
- Neefektyvumas tam tikromis sąlygomis
- Platformos apribojimai

---

# Klausimai ir diskusijos

- Kur dar galite pritaikyti rekursiją savo gyvenime ar kituose mokomuosiuose dalykuose?

---

# Užduotis namuose

- Sugalvoti problemą, kurią galima būtų išspręsti naudojant rekursiją, ir parašyti funkciją

# Daugiau

- [https://inf-knyga.nmakademija.lt/lt/latest/04\\_rekursija.html#f12](https://inf-knyga.nmakademija.lt/lt/latest/04_rekursija.html#f12)
- <https://algor.dev/en/recursion-what-is-it/>
- [https://en.wikipedia.org/wiki/Recursion\\_\(computer\\_science\)](https://en.wikipedia.org/wiki/Recursion_(computer_science))
- <https://blog.skillfactory.ru/glossary/rekursiya/>
- <https://practicum.yandex.ru/blog/rekursiya-v-programmirovanii/>





Pabaiga