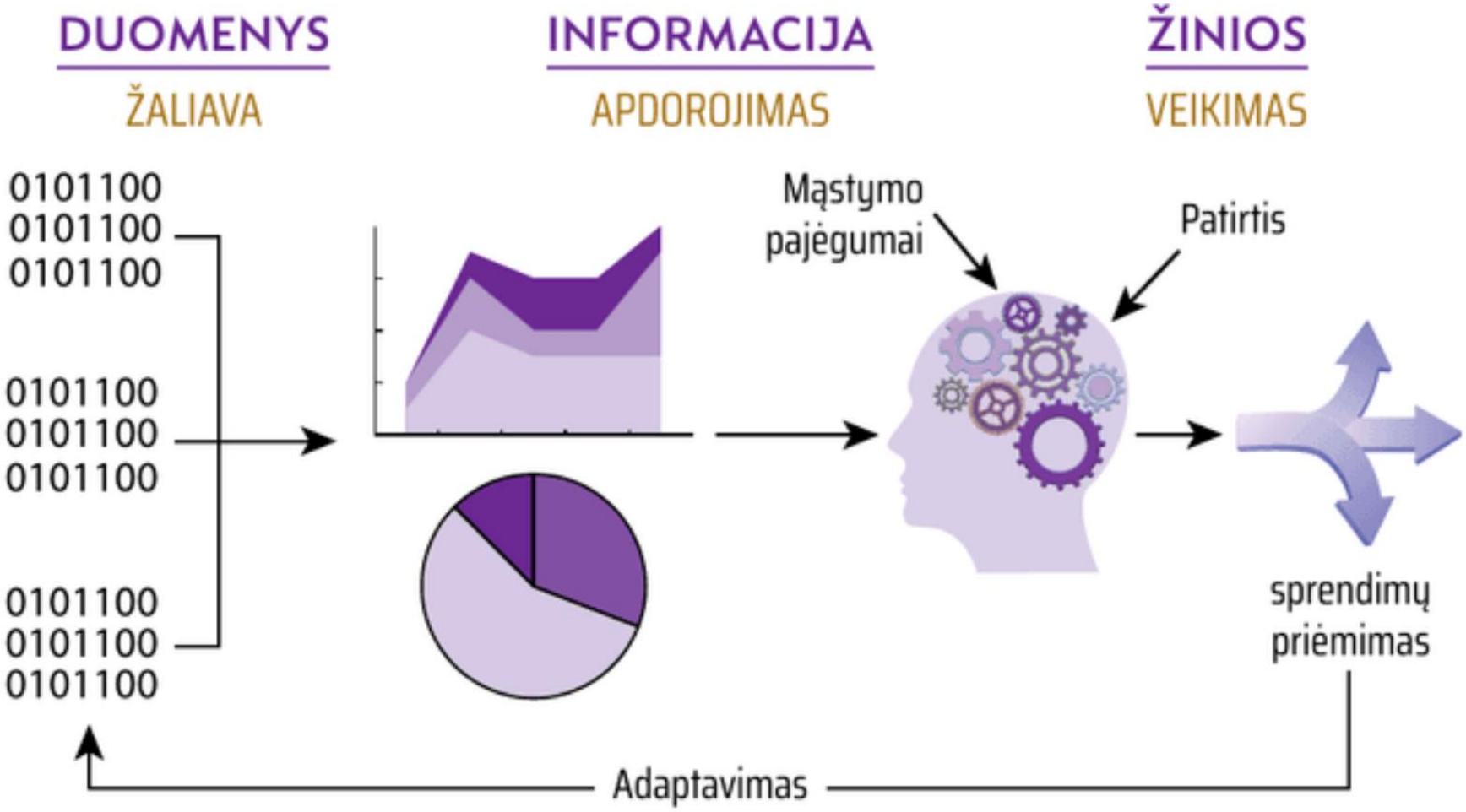
Duomenų struktūrų naudojimas.

Praėjusios pamokos santrauka

- Rekursijos sąvoka ir principai
- Mengerio kempinė
- Sierpinskio trikampis
- Praktinė dalis:
  - faktorialo skaičiavimas
  - Fibonačio seka.
- Rekursijos privalumai ir trūkumai
- Žaidimas „Atspėk skaičių per 10 bandymų”



Šaltinis: *INFORMACINĖS TECHNOLOGIJOS Vadovėlis, Bitas. 9 klasė*

Namų darbų tikrinimas

Sugalvoti problemą, kurią galima būtų išspręsti naudojant rekursiją, ir parašyti funkciją.
Rekursijos algoritmų pavyzdžiai programavime:

https://practicum.yandex.ru/blog/rekursiya-v-programmirovanii/

Tikslai

- Suprasti pagrindines duomenų struktūrų sąvokas
  - masyvai
  - susietieji sąrašai
  - stekai / deklai (dekas)
  - eilės
- Įgyvendinimas naudojant Python ir C++
- Suprasti įvairių duomenų struktūrų privalumus ir trūkumus

Kas yra duomenų struktūros?

Duomenų struktūros – tai būdai, kaip organizuoti, saugoti ir manipuliuoti duomenimis kompiuteryje.
Jų naudojimas leidžia efektyviai valdyti didelius duomenų kiekius, optimizuoti paieškas, įterpimus, ištrynimus bei atlikti kitus veiksmus.

Kodėl jos svarbios

- Pagerina programų veikimo greitį ir efektyvumą.
- Leidžia pasirinkti tinkamiausią duomenų valdymo būdą konkrečiai užduočiai.
- Suteikia lankstumo dirbant su sudėtingais duomenų tipais ir struktūromis.

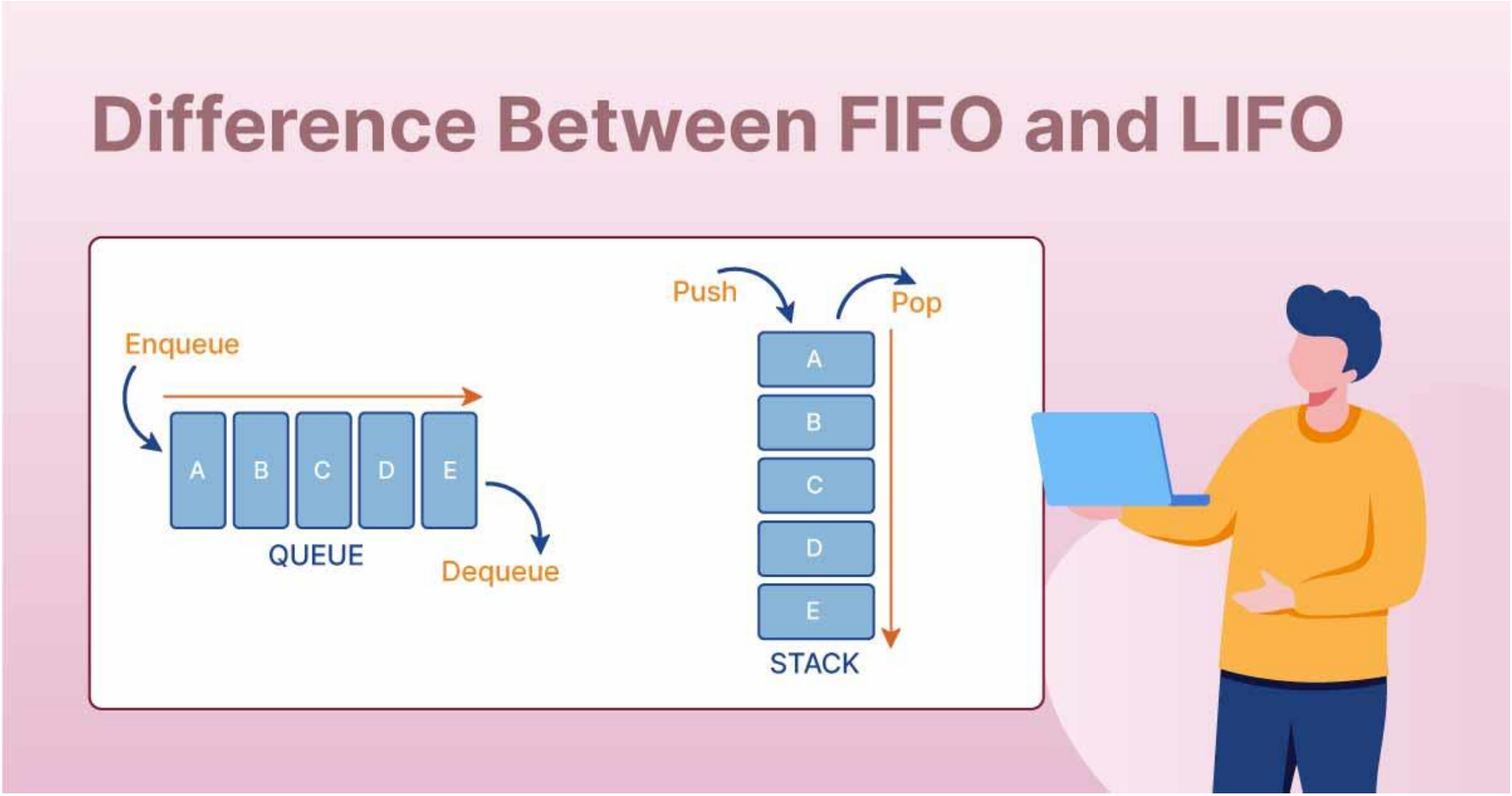
Palyginimas su realiu pasauliu

Masyvai: Kaip knygų lentynos, kuriose kiekviena knyga turi savo numerį (indeksą).

Eilės: Kaip eilė banke, kur pirmas atėjęs yra pirmas aptarnaujamas.

Stekai: Kaip lėkščių krūva – pirma išimama ta, kuri buvo padėta paskutinė.

FIFO (First In, First Out)
LIFO (Last In, First Out)



Šaltinis: https://www.shiksha.com/online-courses/articles/difference-between-fifo-and-lifo/

Stekas (Stack)

- Stekas yra duomenų struktūra, kuri veikia pagal principą **LIFO** (Last In, First Out). Tai reiškia, kad paskutinis į steką įdėtas elementas bus pirmas iš jo pašalintas.
- Operacijos:
  - push() – įdeda elementą į steko viršų.
  - pop() – pašalina elementą iš steko viršaus.
  - peek() – peržiūri, koks elementas yra steko viršuje, jo nepašalinant.

Deklas (Deque)

- Deklas yra duomenų struktūra, kuri veikia kaip dvipusė eilė (double-ended queue), leidžianti pridėti ir pašalinti elementus iš abiejų eilių galų.
- Deklas leidžia tiek **FIFO** (First In, First Out), tiek **LIFO** veikimą, priklausomai nuo to, iš kurio galo pridedami ar pašalinami elementai.
- Operacijos:
  - push_front() / push_back() – įdeda elementą į priekį arba galą.
  - pop_front() / pop_back() – pašalina elementą iš priekio arba galo.

Steko ir deklas pagrindiniai skirtumai

- Stekas (angl. *stack*)
- Deklas (dekas) (angl. *deque*, *double-ended queue*)

- **Stekas** yra griežtai apribotas tik vienu galu: galite pridėti ar pašalinti elementus tik iš steko viršaus.
- **Deklas** yra lankstesnis: leidžia pridėti ir pašalinti elementus tiek iš priekio, tiek iš galo.

Stekas Python ir C++ kalbose

- Python kalboje nėra specialios standartinės bibliotekos stekui, tačiau list tipo objektas gali būti naudojamas kaip stekas, nes jis palaiko **LIFO** (Last In, First Out) principą, naudojant append() ir pop() metodus.
- C++ kalboje stekas yra realizuotas kaip dalis STL (Standard Template Library) ir naudojamas per std::stack. Ši duomenų struktūra yra specialiai sukurta **LIFO** principu, o jos veikimas remiasi tokiais konteineriais kaip std::deque, std::vector, ar std::list

```python
from collections import deque

stack = deque()

# Pridėjimas ir pašalinimas naudojant deque
stack.append(10)
stack.append(20)
stack.pop()     # Pašalina paskutinį elementą (20)
stack.pop()     # Pašalina paskutinį elementą (10)
```

```cpp
#include <iostream>
#include <stack>

int main() {
    std::stack<int> stack;

    // Pridedame elementus į steką
    stack.push(10);
    stack.push(20);

    // Pašaliname elementus iš steko
    stack.pop();    // Pašalina paskutinį elementą (20)
    stack.pop();    // Pašalina paskutinį elementą (10)

    // Patikriname, ar stekas tuščias
    if (stack.empty()) {
        std::cout << "Stekas yra tuščias" << std::endl;
    }

    return 0;
}
```

deque gali būti efektyvesnis už list dėl greitesnio elementų pridėjimo ir pašalinimo iš abiejų galų
deque gali būti efektyvesnis už list dėl greitesnio elementų pridėjimo ir pašalinimo iš abiejų galų

Steko naudojimas Python kalboje

- Python kalboje stekas gali būti realizuotas naudojant du metodus:
  - list – dažniausiai naudojamas steko realizacijai.
  - collections.deque – efektyvesnė alternatyva, skirta darbui su steku, nes leidžia greičiau atlikti elementų pridėjimą ir pašalinimą.

```python
# Naudojame list kaip steką
stack = []

# Pridedame elementus į steką
stack.append(10)
stack.append(20)

# Pašaliname elementus iš steko
print(stack.pop())  # Pašalina ir išveda paskutinį elementą (20)
print(stack.pop())  # Pašalina ir išveda paskutinį elementą (10)
```

```python
from collections import deque

stack = deque()

# Pridėjimas ir pašalinimas naudojant deque
stack.append(10)
stack.append(20)
stack.pop()     # Pašalina paskutinį elementą (20)
stack.pop()     # Pašalina paskutinį elementą (10)
```

deque gali būti efektyvesnis už list dėl greitesnio elementų pridėjimo ir pašalinimo iš abiejų galų

Pagrindinės list operacijos Python kalboje

- **append(x)** – prideda elementą į sąrašo pabaigą.
- **insert(i, x)** – prideda elementą į nurodytą poziciją.
- **remove(x)** – pašalina pirmąją nurodyto elemento reikšmę sąraše.
- **pop(i)** – pašalina ir grąžina elementą iš nurodytos pozicijos. Jei pozicija nepateikta, pašalina paskutinį elementą.
- **clear()** – pašalina visus sąrašo elementus.
- **sort()** – rikiuoja sąrašą vietoje pagal numatytąją arba pasirinktą funkciją.
- **reverse()** – apverčia sąrašo elementų eilę.
- **index(x)** – grąžina pirmo nurodyto elemento indeksą.
- **count(x)** – grąžina, kiek kartų elementas atsiranda sąraše.
- **extend()** – prideda visus kito sąrašo elementus į esamo sąrašo pabaigą.
- **+** operatorius – sujungia du sąrašus ir grąžina naują sąrašą.
- **len()** – grąžina sąrašo elementų skaičių.
- **copy()** – sukuria paviršinę sąrašo kopiją.

Pagrindinės stack operacijos C++ kalboje

- **push(x)** – prideda elementą į steką.
- **pop()** – pašalina paskutinį elementą.
- **top()** – grąžina paskutinį elementą (nepašalinant jo).
- **empty()** – patikrina, ar stekas tuščias.

Palyginimas

- Python: Naudojant list ar deque, stekas yra lankstus ir efektyvus. deque yra efektyvesnis, ypač kai reikia dažnai pridėti ar pašalinti elementus iš abiejų galų.
- C++: std::stack yra specializuota steko klasė, kuri užtikrina saugų ir efektyvų **LIFO** veikimą, naudodama tokias struktūras kaip deque, vector ar list.

Abi kalbos suteikia paprastus ir efektyvius būdus realizuoti steko operacijas, naudojant standartines bibliotekas.

Deklas (Deque) Python ir C++ kalbose

- Python standartiškai turi deklą (deque), kuris yra dalis collections bibliotekos.

```python
# Python

from collections import deque

# Sukuriame tuščią deklą
d = deque()

# Pridedame elementus į galą ir į priekį
d.append(10)        # Prideda į galą
d.appendleft(20)    # Prideda į priekį

# Pašaliname elementus iš galo ir priekio
d.pop()             # Pašalina iš galo
d.popleft()         # Pašalina iš priekio

print(d)  # Išveda deque([])
```

- C++ turi standartinę deklą (*deque*), kuris yra dalis STL (Standard Template Library). Jis yra optimizuotas naudoti su abiejų galų operacijomis.

```cpp
// C++

#include <iostream>
#include <deque>

int main() {
    std::deque<int> d;

    // Pridedame elementus į galą ir į priekį
    d.push_back(10);    // Prideda į galą
    d.push_front(20);   // Prideda į priekį

    // Pašaliname elementus iš galo ir priekio
    d.pop_back();       // Pašalina iš galo
    d.pop_front();      // Pašalina iš priekio

    // Patikriname, ar deklas tuščias
    if (d.empty()) {
        std::cout << "Deklas yra tuščias." << std::endl;
    }

    return 0;
}
```

Svarbios operacijos (Python)

- **append(x)** – prideda elementą į deklą gale.
- **appendleft(x)** – prideda elementą į deklą priekyje.
- **pop()** – pašalina ir grąžina paskutinį elementą.
- **popleft()** – pašalina ir grąžina pirmą elementą.
- **extend(iterable)** – prideda iteruotiną objektą (pvz., sąrašą) į deklą gale.
- **extendleft(iterable)** – prideda iteruotiną objektą į deklą priekyje.

Svarbios operacijos (C++)

- **push_back(x)** – prideda elementą į deklą gale.
- **push_front(x)** – prideda elementą į deklą priekyje.
- **pop_back()** – pašalina paskutinį elementą.
- **pop_front()** – pašalina pirmą elementą.
- **at(index)** – pasiekia elementą pagal indeksą.
- **size()** – grąžina deklą dydį.
- **empty()** – patikrina, ar deklas tuščias.

Skirtumai tarp Python ir C++ implementacijų

- **Python:** deque yra optimizuotas naudoti abiem galais ir palaiko saugų, išplėstą valdymą, pvz., limituotą dydį, rotacijas, tačiau yra aukštesnio lygio abstrakcija.
- C++: std::deque yra žemesnio lygio, labiau valdomas našumo požiūriu ir naudoja tipinius STL metodus, leidžiančius atlikti efektyvius veiksmus tiek gale, tiek priekyje.

Abi kalbos turi savo ypatybes, tačiau deklas yra labai efektyvus ir patogus abiem kalbomis.

Pabaiga