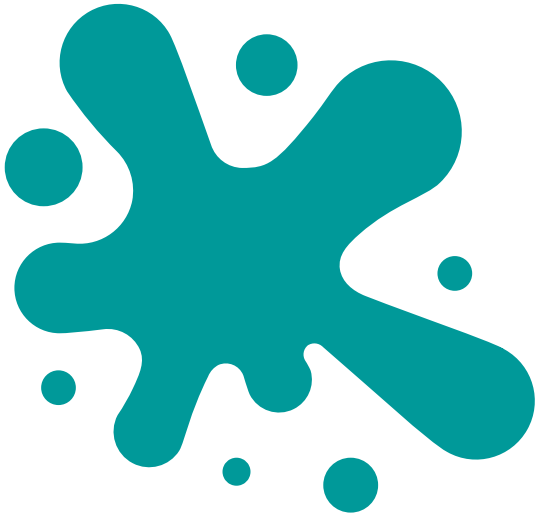

Duomenų struktūrų naudojimas.

Duomenų struktūros

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

Ką išmoksime šiandien

- Suprasime pagrindines duomenų struktūrų sąvokas
- Suprasime sąrašų Python ir masyvų C++ pagrindines sąvokas.
- Išmoksime naudoti sąrašus ir masyvus duomenų saugojimui ir tvarkymui.
- Atliksime praktines užduotis, taikydami šias duomenų struktūras Python ir C++ programose.

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.

Pagrindinės sekos tipo duomenų struktūros

Python List: Dinaminis masyvas, kuris gali keisti savo dydį

C++ Array: Fiksuoto dydžio elementų rinkinys, kur visi elementai turi tą patį tipą

```python
# Dinamiškai plečiama struktūra
my_list = [1, 2, 3, 4, 5]
my_list.append(6)
```

```cpp
// Fiksuoto dydžio struktūra
int my_array[5] = {1, 2, 3, 4, 5};
```

Python sąrašai

- Python sąrašai yra **dinamiški ir kintami** masyvai.
- Sąrašai gali talpinti skirtingų tipų elementus, pvz., skaičius, tekstus ar objektus

Python sąrašai (pagrindinės operacijos)

- Sąrašo sukūrimas

```
my_list = [1, "apple", 3.5, True]
```

- Elementų pasiekimas

```
first_item = my_list[0]  # 1
```

- Pridėjimas

```
my_list.append("new item")
```

- Šalinimas

```
# Pašalina elementą pagal vertę
my_list.remove("apple")

# Pašalina elementą pagal indeksą
my_list.pop(2)
```

- Sąrašo dalijimas (slicing)

```
# ["apple", 3.5]
sub_list = my_list[1:3]
```

Sąrašo dalijimas (slicing)

```
['Obuolys', 'Vyšnios', 'Apelsinas', 'Mango', 'Kivis']
```

```
print(vaisiai[1:3])
['Vyšnios', 'Apelsinas']
```

```
print(vaisiai[2:4])
['Apelsinas', 'Mango']
```

Kodo pavyzdys (Python)

```python
# Sąrašo sukūrimas
vaisiai = ["Obuolys", "Bananai", "Vyšnios"]

# Elementų pasiekimas
print(vaisiai[0])  # Išvestis: Obuolys

# Elementų pridėjimas
vaisiai.append("Apelsinas")
vaisiai.append("Mango")
vaisiai.append("Kivis")

# Elementų šalinimas
vaisiai.remove("Bananai")

print(vaisiai)

# Sąrašo dalijimas
print(vaisiai[1:3])  # Išvestis: ['Vyšnios', 'Apelsinas']
print(vaisiai[2:4])  # Išvestis: ['Apelsinas', 'Mango']
```

Pagrindinės „list“ operacijos (Python)

- **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.

Masyvai C++

- C++ masyvai yra **fiksuoto dydžio** elementų rinkiniai
- Visi masyvo elementai turi būti **to paties tipo** (pvz., visi skaičiai arba visi simboliai)
- Masyvo dydis nustatomas sukūrimo metu ir negali būti keičiamas vėliau

Masyvai C++ (pagrindinės operacijos)

- Masyvo sukūrimas

```
// Masyvas su 5 sveikaisiais skaičiais
int my_array[5] = {1, 2, 3, 4, 5};
```

- Elementų pasiekimas

```
// 1-asis elementas (1)
int first_element = my_array[0];
```

- Reikšmių keitimas

```
// 3-asis elementas dabar yra 10
my_array[2] = 10;
```

Kodo pavyzdys (C++)

```cpp
#include <iostream>
using namespace std;

int main() {
    // Masyvo sukūrimas
    int skaiciai[5] = {1, 2, 3, 4, 5};

    // Elementų pasiekimas
    cout << "Pirmas elementas: " << skaiciai[0] << endl;

    // Elementų keitimas
    skaiciai[2] = 10;
    cout << "Pakeistas trečias elementas: " << skaiciai[2] << endl;

    // Masyvo perėjimas ciklu
    for(int i = 0; i < 5; i++) {
        cout << skaiciai[i] << " ";
    }

    return 0;
}
```

Praktinė užduotis (Python)

- Python
  - Sukurkite savo Python sąrašą su 4 mėgstamais maisto produktais ir jį modifikuoti naudojant 2 append ir 3 remove. Atvaizduokite rezultatą. (List_02.py)

- C++
  - Sukurkite C++ masyvą su 6 mėgstamais skaičiais ir pakeisti trečiąjį ir paskutinį elementus. Atvaizduokite rezultatą.

Sąrašai vs Masyvai

- **Lankstumas:** Python sąrašai yra dinamiški, C++ masyvai – fiksuoti.
- **Duomenų tipai:** Python sąrašai gali talpinti skirtingus duomenų tipus, o C++ masyvai – tik vieno tipo elementus.
- **Atmintis:** Python sąrašai automatiškai keičia dydį, o C++ masyvai fiksuoja atmintį nuo pradžios.

Apibendrinimas

- Susipažinome su duomenų struktūrų sąvoka
- Susipažinome su sąrašų Python ir masyvų C++ pagrindinėmis sąvokomis.
- Išmokome naudoti sąrašus ir masyvus duomenų saugojimui ir tvarkymui.
- Atlikome praktines užduotis, taikydami šias duomenų struktūras Python ir C++ programose.

Namų darbas

Pasirinkite vieną iš šių dviejų variantų:

- Parašykite **Python** programą, kuri naudoja sąrašą savaitės temperatūroms sekti ir apskaičiuoja vidutinę temperatūrą. NEGALIMA naudoji sum() ar analogiškos funkcijos.

- Parašykite **C++** programą, kuri naudoja masyvą 5 egzaminų pažymiams saugoti ir pateikia aukščiausią pažymį. NEGALIMA naudoji funkcijos, kurį apskaičiuoja visų masyvo elementų sumą.

Pabaiga