# Algoritmų parinkimas.

Ciklai. „While“ ciklas

# While ir For ciklų skirtumai

- **Naudojimo paskirtis:**

  - **While ciklas:** Naudojamas, kai nežinome, kiek kartų reikės kartoti ciklą iš anksto. Jis vykdomas tol, kol sąlyga yra teisinga.

    **Pavyzdys:** Kol vartotojas įveda teisingą slaptažodį, kartoti prašymą įvesti slaptažodį.

  - **For ciklas:** Naudojamas, kai iš anksto žinome, kiek kartų norime kartoti ciklą, arba norime iteruoti per elementus (pvz., per sąrašą ar masyvą).

    **Pavyzdys:** Iteruoti per sąrašą elementų arba skaičiuoti nuo 1 iki 10.

# While ir For ciklų skirtumai (2)

- Sintaksė:

```python
# Python

while sąlyga:
    # kodas vykdomas tol, kol sąlyga teisinga

for kintamasis in seka:
    # kodas vykdomas per kiekvieną elemento iteraciją
```

```cpp
// C++

while (sąlyga) {
    // kodas vykdomas tol, kol sąlyga teisinga
}

for (inicializacija; sąlyga; inkrementas) {
    // kodas vykdomas nurodytą kartų skaičių
}
```

# While ir For ciklų skirtumai (3)

- While ciklas:
  - Reikia ***rankiniu būdu kontroliuoti***, kada sąlyga taps klaidinga, t. y., patys turime atnaujinti kintamąjį ciklo viduje, kad ciklas nutrūktų.

- For ciklas:
  - Įtraukia inicializavimą, sąlygą ir atnaujinimą į vieną eilutę, todėl ***ciklo kintamasis valdomas automatiškai***.

# While ir For ciklų skirtumai (4)

- **While ciklas:** Kintamasis i inicializuojamas už ciklo ribų ir didinamas ciklo viduje. Ciklas kartojamas tol, kol sąlyga i < 5 yra teisinga.
- **For ciklas:** *Inicializavimas* (int i = 0), *sąlyga* (i < 5), ir *inkrementavimas* (i++) yra parašyti vienoje eilutėje. Ciklas vyksta tol, kol sąlyga yra teisinga.

```python
# Python

i = 0
while i < 5:
    print(i)
    i += 1

for i in range(5):
    print(i)
```

```cpp
// C++

int i = 0;
while (i < 5) {
    cout << i << endl;
    i++;
}

for (int i = 0; i < 5; i++) {
    cout << i << endl;
}
```

# While ir For ciklų skirtumai (5)

- While ciklas:
  - Kai reikia kartoti tol, ***kol tenkinama tam tikra sąlyga***, bet nežinome, kiek kartų ciklas bus vykdomas.
- For ciklas:
  - Kai ***ciklas turi aiškų kartų skaičių*** arba kai reikia iteruoti per masyvus, sąrašus ar kitas struktūras.

# Sakinys „break“ (while/for)

- **Tikslas:** Naudojant break sakinį galime nutraukti ciklo vykdymą, net jei while/for sąlyga dar teisinga.
- **Veikimas:** Kai programa pasiekia break, ciklas iškart nutraukiamas ir programa tęsiasi už ciklo ribų.

```python
# Python

i = 1
while i < 6:
    print(i)
    if i == 3:
        break
    i += 1
```

```cpp
// C++

int i = 1;
while (i < 6) {
    cout << i << endl;
    if (i == 3) {
        break;
    }
    i++;
}
```

# Sakinys „continue“ (while/for)

- **Tikslas:** Naudojant continue sakinį galime praleisti dabartinę ciklo iteraciją ir iškart pereiti prie kitos.
- **Veikimas:** Kai programa pasiekia continue, ciklo likusi dalis praleidžiama, o vykdymas pereina prie kitos iteracijos.

```python
i = 0
while i < 6:
    i += 1
    if i == 3:
        continue
    print(i)
```

```cpp
// C++

int i = 0;
while (i < 6) {
    i++;
    if (i == 3) {
        continue;
    }
    cout << i << endl;
}
```

# Sakinys „else“ (while/for) – Python

- **Tikslas:** else sakinys cikluose naudojamas vykdyti kodą, kai ciklas baigiasi natūraliai, t. y. nesustabdomas per break.
- **Veikimas:** else blokas įvykdomas, kai sąlyga cikle tampa klaidinga arba kai ciklo iteracijos baigiasi.

```
i = 0
while i < 3:
    print(i)
    i += 1
else:
    print("Ciklas baigtas")

for i in range(3):
    print(i)
else:
    print("Ciklas baigtas")
```

Svarbu: else blokas nebus vykdomas, jei ciklas nutraukiamas naudojant break.