Paprogramės

Parametrai

Praėjusios pamokos santrauka

- Funkcija range() (Python)
- Sakinys „pass“
- Ciklas cikle (nested loops)

Ką išmoksime šiandien

- Supažinsime su paprogramių (angl. functions) sąvoka.
- Išmokysime kurti ir naudoti paprogrames su parametrais.
- Suprasime kaip veikia parametrai, perduodami į paprogrames, bei skirtumus tarp vertės ir nuorodos perdavimo.
- Atliksime praktinius užsiėmimus su paprogramėmis Python ir C++ kalbose.

Kas yra paprogramės?

Paprogramė – tai kodas, kuris atlieka konkrečią užduotį ir gali būti kviečiamas iš bet kurios programos vietos.

Paprogramių privalumai

Kodėl naudojamos paprogramės?

- Leidžia išskaidyti sudėtingą programą į mažesnes, lengviau valdomas dalis.
- Pagerina kodo struktūrą ir skaitymo aiškumą.
- Sumažina kodo dubliavimą.
- Padeda lengviau derinti ir išlaikyti programinę įrangą.

Pavyzdys iš kasdienio gyvenimo

Maisto gaminimo receptas – tai paprogramė:

- Nuoseklios instrukcijos, vedančios nuo pradžios iki rezultato.
- Gali būti naudojamos daug kartų, skirtingomis situacijomis.

Paprogramės kūrimo sintaksė (Python)

```
def function_name(parameters):
    # Function body
    return result
```

- def – žodis, naudojamas paprogramės apibrėžimui.
- Po jo nurodomas paprogramės pavadinimas ir parametrai skliausteliuose.

```
def greet(name):
    print("Hello, " + name)
```

Paprogramės kūrimo sintaksė

```python
def function_name(parameters):
    # Function body
    return result
```

```cpp
return_type function_name(parameters) {
    // Function body
    return result;
}
```

Grąžinimo reikšmė (Python)

return: Grąžinimo reikšmė

```
def add_numbers(a, b):
    return a + b
```

- Paprogramė gali grąžinti reikšmę, kurią vėliau galima naudoti kitur programoje.
- return – komanda, naudojama reikšmei grąžinti

Parametrai (Python)

Parametrai paprogramėje

```
def multiply(a, b):
    return a * b
```

- Parametrai – tai kintamieji, kuriuos galima perduoti paprogramei, kad ji veiktų su skirtingomis reikšmėmis.
- Parametrus rašome skliausteliuose po paprogramės pavadinimo.

Grąžinimo tipas ir funkcijos apibrėžimas (C++)

Funkcijos grąžinimo tipas

- **Grąžinimo tipas** nurodo, kokio tipo reikšmę paprogramė grąžins (pvz., int, float, void).
- Po to seka paprogramės pavadinimas ir parametrai skliausteliuose.
- Paprogramės apibrėžiamos naudojant kūrimo blokus {}.

```
int add_numbers(int a, int b) {
    return a + b;
}
```

Grąžinimo reikšmė

return: Grąžinimo reikšmė

- Paprogramė grąžina reikšmę su return. Ši reikšmė gali būti bet kokio tipo, nurodyto apibrėžiant funkciją.

```
float area(float radius) {
    return 3.14 * radius * radius;
}
```

Parametrai

Parametrai paprogramėje

- Parametrai perduodami paprogramei kvietimo metu. Jie nurodomi skliausteliuose po paprogramės pavadinimo.
- Parametrai leidžia paprogramėms dirbti su skirtingomis reikšmėmis kiekvieną kartą, kai jos yra kviečiamos.

```
void print_sum(int a, int b) {
    cout << "Sum: " << a + b << endl;
}
```

Parametrai su numatytomis reikšmėmis

- **Python**: parametrai gali turėti numatytas reikšmes.

```python
def greet(name="student"):
    print("Hello, " + name)
```

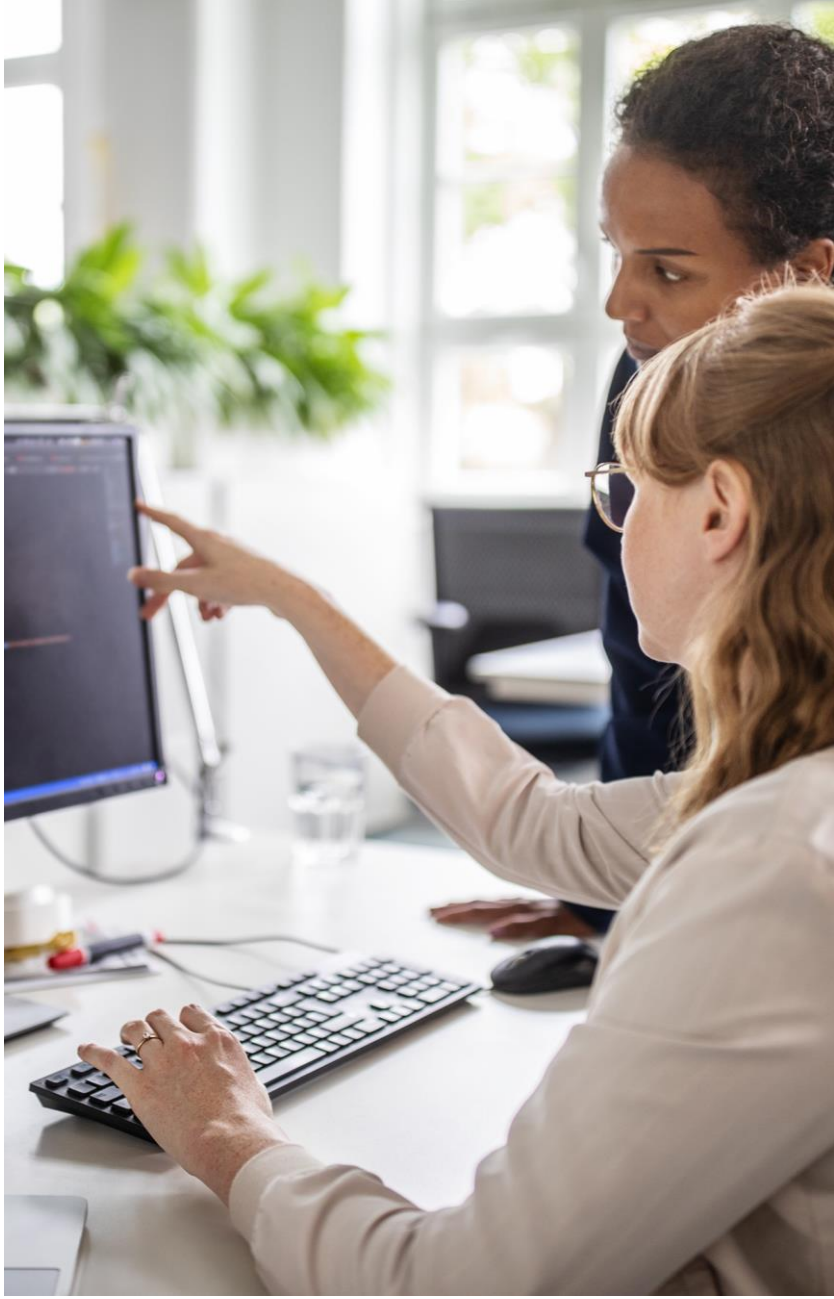
- **C++**: numatytos reikšmės taip pat galimos.

```cpp
void greet(string name = "student") {
    cout << "Hello, " + name << endl;
}
```

Demonstravimas (**Python**): paprastos funkcijos sukūrimas, kuri grąžina dviejų skaičių sumą

```
def add_numbers(a, b):
    return a + b

print(add_numbers(3, 5))
```



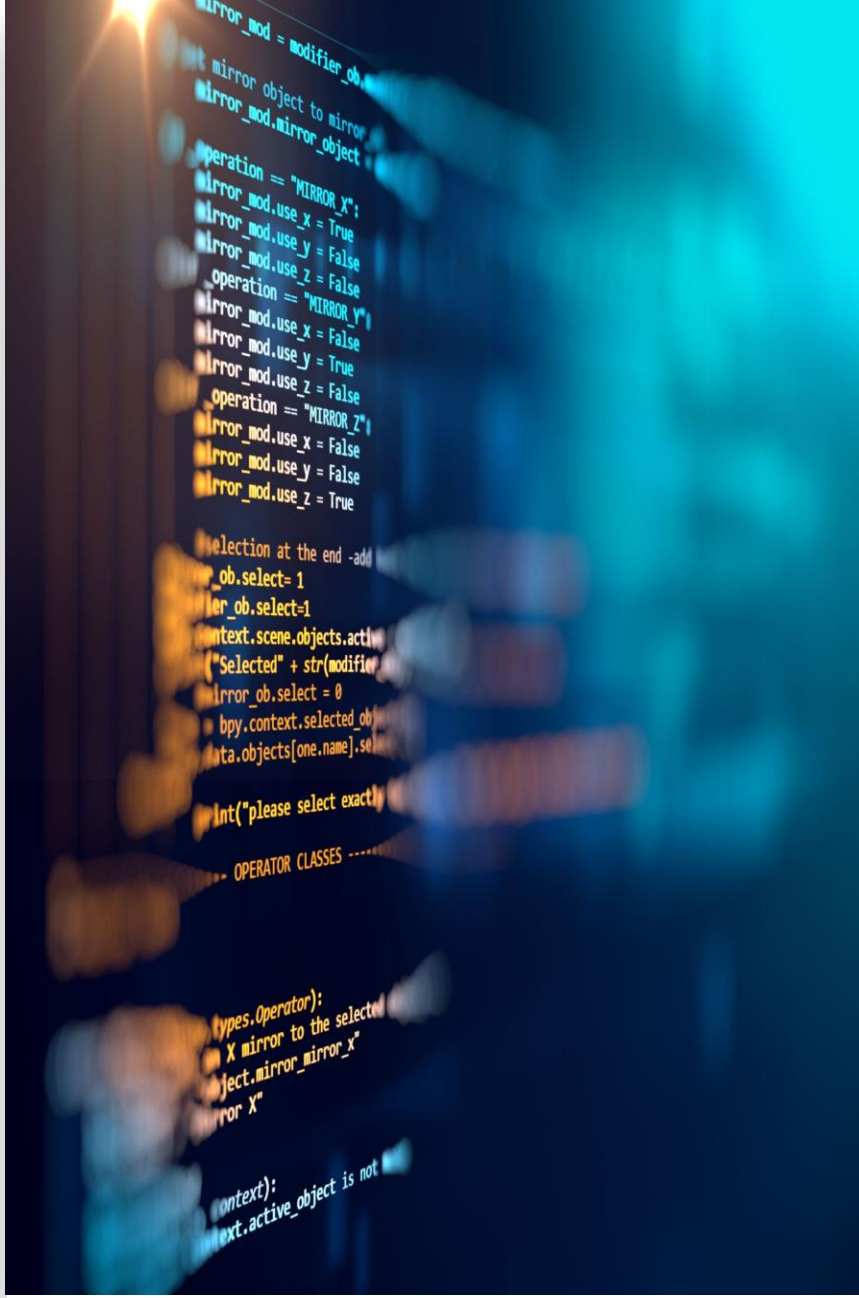
https://github.com/agorbatniov/inf11/blob/main/Paprogrames/Python/Paprogrames_02_demo.py

Demonstravimas (**C++**): paprogramės su int parametrais ir grąžinimo reikšme

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

int add_numbers(int a, int b) {
    return a + b;
}

int main() {
    cout << add_numbers(3, 5) << endl;
    return 0;
}
```



https://github.com/agorbatniov/inf11/blob/main/Paprogrames/C++/Paprogrames_02_demo.cpp

Praktinė užduotis (5 min)

Paprastos funkcijos sukūrimas, kuri apskaičiuoja dviejų skaičių kvadrato sumą (Python ir C++)

Parametrų perdavimas

- **Pagal vertę:** į paprogramę perduodama parametrų reikšmė. Bet koks parametrų pakeitimas paprogramės viduje neatsispindi išoriniame kode.

- **Pagal nuorodą:** perduodama tiesioginė nuoroda į atminties vietą, todėl paprogramė gali keisti pradinio parametro reikšmę.

Perdavimas pagal vertę

- Naudojamas dažniausiai, kai nereikia keisti originalių duomenų.
- Kiekvienas parametras kopijuojamas į paprogramę, ir bet kokie pakeitimai lieka tik funkcijos viduje.

```
void add(int a) {
    a += 10;
}
```

Perdavimas pagal nuorodą

- Naudojamas, kai reikia keisti originalią reikšmę arba išvengti didelių duomenų kopijavimo.
- Paprogramė gali tiesiogiai keisti perduodamo parametro reikšmę.

```
void add(int &a) {
    a += 10;
}
```

Python parametrai: tik vertės perdavimas

- Python visi parametrai perduodami **pagal vertę** (angl. "pass-by-value").
- Paprogramėje gaunama parametro kopija, todėl originali reikšmė nėra keičiama, kai perduodamos bazinės duomenų struktūros.

```
def modify(x):
    x += 10
    return x
```

- Tačiau perduodant **sudėtines struktūras**, kaip sąrašai, originali struktūra **gali būti modifikuota**, nes kopijuojama tik nuoroda į objektą

```
def modify_list(my_list):
    my_list.append(10)
```

Praktinė užduotis

Sukurti paprogramę, kuri apskaičiuoja apskritimo plotą.

- Parametras: apskritimo spindulys.

Papildoma užduotis:

- Pridėti numatytą reikšmę spinduliui, jei jis nėra perduotas.

Apskritimo ploto formulė:

Plotas=$\pi \times r2$, kur r – apskritimo spindulys.

```
import math

math.pi
```

```
#include <cmath>

M_PI;
pow(x, y);
```

Apibendriinimas

- Supažinome su paprogramių (angl. functions) sąvoka.
- Išmokome kurti ir naudoti paprogrames su parametrais.
- Supratome kaip veikia parametrai, perduodami į paprogrames, bei skirtumus tarp vertės ir nuorodos perdavimo.
- Atlikome praktinius užsiėmimus su paprogramėmis Python ir C++ kalbose.

Pabaiga