Napredno programiranje i programski jezici

02 C++

Fakultet tehničkih nauka, Novi Sad
23-24/Z
Dunja Vrbaški

FUNKCIJE

```
tipPovratneVrednosti nazivFunkcije ( param1, param2, ...) { teloFunkcije }
```

```
int saberi(int x, int y)
{
    return x + y;
}

int main()
{
    cout << saberi(2, 3);
    return 0;
}
```

x, y - formalni parametri // parametri

2, 3 - stvarni parametri // argumenti

```
int main()
{
    cout << saberi(2, 3);
    return 0;
}

int saberi(int x, int y)
{
    return x + y;
}
```

```
error: 'saberi' was not declared in this scope
```

```
int saberi(int, int);

int main()
{
    cout << saberi(2, 3);
    return 0;
}

int saberi(int x, int y)
{
    return x + y;
}
```

main.cpp

```cpp
#include <iostream>
#include "biblioteka.hpp"

using namespace std;

int main()
{
    cout << saberi(2, 3);
    return 0;
}

int saberi(int x, int y)
{
    return x + y;
}
```

biblioteka.hpp

```cpp
int saberi(int, int);
```

main.cpp

```cpp
#include <iostream>
#include "biblioteka.hpp"

using namespace std;

int main()
{
    cout << saberi(2, 3);
    return 0;
}
```

biblioteka.hpp

```cpp
#ifndef BIBLIOTEKA_HPP_INCLUDED
#define BIBLIOTEKA_HPP_INCLUDED

int saberi(int, int);

#endif // BIBLIOTEKA_HPP_INCLUDED
```

guards

biblioteka.cpp

```cpp
int saberi(int x, int y)
{
    return x + y;
}
```

header fajl daje instrukcije šta može i kako da se koristi iz biblioteke, programski interfejs.
source code može biti nedostupan.

```
int saberi(int x = 0, int y = 0)
{
    return x + y;
}

int main()
{
    cout << saberi(2, 3);
    cout << saberi();
    return 0;
}
```

podrazumevane vrednosti parametara
opcioni parametri

ovde, u primerima gde suštinski nije važno, ne razdvajamo u posebnu biblioteku

```
int saberi(int x, int y = 0)
{
    return x + y;
}

int main()
{
    cout << saberi(2, 3);
    cout << saberi(2);
    return 0;
}
```

```
int saberi(int x, int y = 0)
{
    return x + y;
}

int saberi(int x = 0, int y)
{
    return x + y;
}

int main()
{
    cout << saberi(2, 3);
    cout << saberi(2);
    return 0;
}
```

```
error: default argument missing for parameter 2 of
'int saberi(int, int)'|
```

```
int saberi(int x, int y)
{
    return x + y;
}

int saberi(int x, int y, int z)
{
    return x + y + z;
}

int main()
{
    cout << saberi(2, 3);
    cout << saberi(2, 3, 4);
    return 0;
}
```

Mogu imati isto ime sve dok se lista parametara razlikuje (broj i tip)

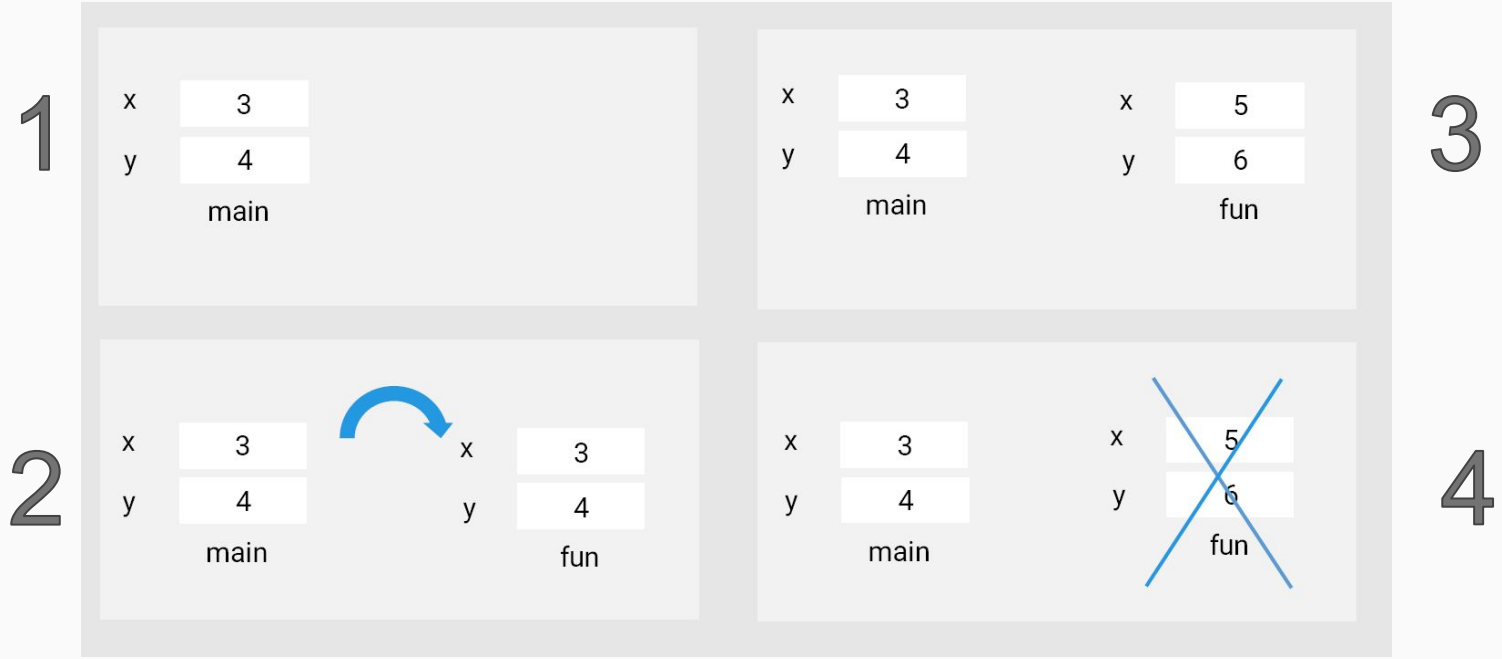
```
int saberi(int x, int y)
{
    return x + y;
}

double saberi(double x, double y)
{
    return x + y;
}

int main()
{
    cout << saberi(2, 3);
    cout << saberi(2.1, 3.2) << endl;
    return 0;
}
```

```
int saberi(int x, int y)
{
    return x + y;
}

double saberi(in x, int y)
{
    return (x + y) * 1.0;
}

int main()
{
    cout << saberi(2, 3);
    return 0;
}
```

```
error: ambiguating new declaration of 'double
saberi(int, int)'
```

```
void fun(int x, int y)
{
    x = 5;
    y = 6;
}

int main()
{
    int x = 3, y = 4
    fun(x, y);
    cout << x << y;
    return 0;
}
```

Šta će biti ispisano?



Prenos po vrednosti

```
void fun(int x, int y)
{
    x = 5;
    y = 6;
}

int main()
{
    int x = 3, y = 4
    fun(x, y);
    cout << x << y;
    return 0;
}
```

Stvarni parametri (argumenti) funkcije se prenose “po vrednosti”.

Pravi se kopija u memorijskom prostoru koji odgovara pozivu funkcije.

Promene nad kopijom ne menjaju originalnu promenljivu.

```
void fun(int x, int y)
{
    x = 5;
    y = 6;
}
void funPok(int *x, int *y)
{
    (*x) = 5;
    (*y) = 6;
}
int main()
{
    int x = 3, y = 4;
    fun(x, y);
    cout << x << y;
    funPok(&x, &y);
    cout << x << y;
    return 0;
}
```

Šta će biti ispisano?

```
void fun(int x, int y)
{
    x = 5;
    y = 6;
}
void funPok(int *x, int *y)
{
    (*x) = 5;
    (*y) = 6;
}
int main()
{
    int x = 3, y = 4;
    fun(x, y);
    cout << x << y;
    funPok(&x, &y);
    cout << x << y;
    return 0;
}
```

Čak i kad su parametri pokazivači i to se prenosi po vrednosti samo što ta vrednost (*vrednost je adresa*) i dalje pokazuje na isti podatak.

Kad koristimo pokazivače i dalje je prenos po vrednosti, ne po referenci. Samo deluje da se radi o prenosu po eferenci. Nekad kažu "prenos po adresi".

Prenos po vrednosti može biti nezgodan kada su podaci veliki.
Puno memorije se kopira.
Uvek možemo iskoristiti pokazivač za prenos, ali moramo paziti da ne promenimo njegovu vrednost (adresu) (`p = nekaNovaAdresa`)

OOP - Rad sa objektima - veliki podaci

→ Prenos po referenci

```
void fun(int x)
{
    x++;
}
void funPok(int *x)
{
    (*x)++;
}
void funRef(int &x)
{
    x++;
}
int main()
{
    int x = 3;
    funRef(x);
    cout << x;
    return 0;
}
```

REFERENCA

Alijas za x
funkcija direktno pristupa originalu

!= pokazivač
nije podatak, samo njegovo drugo ime

kako je implementirano? zavisi od kompajlera

```
void fun(int x)
{
    x++;
}
void funPok(int *x)
{
    (*x)++;
}
void funRef(int &x)
{
    x++;
}
int main()
{
    int x = 3;
    fun(x);     cout << x;
    funPok(&x); cout << x;
    funRef(x);  cout << x;
    return 0;
}
```

Obratiti pažnju na argumente.

Šta će biti ispisano?

```
void fun(int x)
{
    x++;
}
void fun(int *x)
{
    (*x)++;
}
void fun(int &x)
{
    x++;
}
int main()
{
    int x = 3;
    fun(x);  cout << x;
    fun(&x); cout << x;
    fun(x);  cout << x;
    return 0;
}
```

Gde je problem?

```
void fun(int x)
{
    x++;
}
void fun(int *x)
{
    (*x)++;
}
void fun(int &x)
{
    x++;
}
int main()
{
    int x = 3;
    fun(x);  cout << x;
    fun(&x); cout << x;
    fun(x);  cout << x;
    return 0;
}
```

```
error: call of overloaded 'fun(int&)' is
ambiguous|
```

moraju imati drugačije ime

OOP