



PROGRAMIRANJE · SIU 1: Funkcije i rekurzija · Ishod I5 – sat1

Analiza problema i algoritma, preopterećenje funkcija

Analiza ulaza, izlaza, ograničenja i međuzavisnosti

Prijedlog funkcionalnih cjelina prije pisanja koda

- **Prepoznati ulaze, izlaze**, ograničenja i odnose među koracima rješenja.
- **Predložiti funkcije s jasnim odgovornostima, parametrima i povratnim vrijednostima.**
- **Objasniti redoslijed poziva i prijenos rezultata među funkcijama.**
- **Provjeriti algoritam smislenim testovima** i argumentirati izmjenu.
- Prepoznati **valjano preopterećenje po broju ili tipovima parametara** te odbaciti pokušaj razlikovanja samo po povratnom tipu.

Preopterećenje funkcija – problemski primjer 1

1. Ista operacija, različiti tipovi parametara – odabir većeg broja:

```
int veci(int a, int b) { return (a > b) ? a : b; }
```

```
double veci(double a, double b) { return (a > b) ? a : b; }
```

```
// veci(4, 7) → int verzija; veci(4.5, 7.2) → double verzija.
```

Preopterećenje funkcija – problemski primjer 2

2. Obavijest korisniku – isto ime, ali različiti podaci (string ili bool):

```
void prikazi(string poruka) { cout << "Poruka: " << poruka; }
```

```
void prikazi(bool uspjesno) {
```

```
    cout << (uspjesno ? "Postupak je uspio." : "Postupak nije uspio.");}
```

```
prikazi(string("Spremanje"));
```

```
prikazi(true);
```

```
prikazi(1);
```

Upozorenje: prikazi(1) poziva bool inačicu jer se int pretvara u bool. **Za tekstni literal bez pretvorbe u string, primjerice prikazi("Spremanje"), bira se bool verzija (pretvorba pokazivača u bool).** Zato u primjeru koristimo string objekt ili izričito string("Spremanje").

Preopterećenje funkcija – problemski primjer 3

3. Dostava – isto značenje, različit broj argumenata:

```
double dostava(double osnovnaCijena) { return osnovnaCijena; }
```

```
double dostava(double osnovnaCijena, int km) {
```

```
    return osnovnaCijena + km * 0.5;
```

```
}
```

```
// dostava(4.0) → 4;
```

```
dostava(4.0, 10) → 9.
```

Zaključni

- Ključna pravila: preopterećenje se razlikuje po popisu parametara (broj/tipovi).
- Ne preopterećivati funkcije koje rade nepovezane stvari samo zato što to programski jezik dopušta.



PROGRAMIRANJE · SIU 1: Funkcije i rekurzija · Ishod I5 – sat2

Analiza problema i algoritma

Ishodi

- projektirati **potpis, prototipove i tijek poziva**
- promišljeno prepoznati kada je opravdano **preopterećenje funkcija (overloading)**
- Za zadani problem odrediti **funkcije s jednom jasnom odgovornošću, odgovarajućim parametrima i povratnim tipom**
- vizualizirati prijenos vrijednosti među funkcijama
- razlikovati argument od lokalnog parametra.
- Objašnjavati prototip funkcije i usklađenost deklaracije, poziva i definicije.
- Obrazložiti zašto **zajedničko ime funkcija treba označavati konceptualno istu operaciju.**

1. problem - Školski festival: sustav za izračun cijene ulaznica

- Osnovna cijena ulaznice je 12 €.
- Učenik ima 25 % popusta
- Grupa od najmanje 5 osoba dodatnih 10 % na već umanjenu cijenu
- Ukupna cijena = konačna cijena jedne ulaznice $\times$ broj osoba
- Ne dopustiti negativan broj

- Izračunati cijenu za osobu.
- Izračunati ukupnu cijenu za grupu.

ANALIZA i projektiranje funkcija

Model C++ koda

```
double cijenaPoOsobi(bool ucenik, int brojOsoba) {  
    double cijena = 12.0;  
    if (ucenik) cijena *= 0.75;  
    if (brojOsoba >= 5) cijena *= 0.90;  
    return cijena;  
}
```

// Napomena: grananje je dio modela problema;

Koja je odgovornost funkcije?

Koje podatke prima?

Što vraća?

Pitanja

1. Koji podatak funkcija cijenaPoOsobi mora dobiti da bi znala primijeniti grupni popust?

Status učenika i broj osoba (odnosno već utvrđeno pravo na popust).

2. Zašto '*najprije ukupna cijena, zatim popust*' ne odgovara nužno zadanim pravilima?

Pravila nalažu redoslijed popusta na cijenu po osobi; drukčiji redoslijed ili zaokruživanje mogu promijeniti rezultat.

3. Koji test otkriva pogrešku na granici grupnog popusta?

4 i 5 osoba.

predložiti drugi valjan dizajn funkcija i usporediti ga prema jasnoći i testabilnosti.

Al može biti kritičar testova i pogrešaka, ne zamjena za razumijevanje.

2. problemski kontekst: pametna punionica bicikala

- Punionica raspolaže s ukupno brojPriključaka (6) priključaka
- čeka brojBicikala (5) bicikala - predviđen izlaz: 5 0 6.
- U jednoj sesiji prihvaća najviše onoliko bicikala koliko ima slobodnih priključaka.
- Za 1–3 prihvaćena bicikla cijena je 1,50 € po biciklu, a za 4 ili više 1,20 €.
- Program treba **prikazati broj prihvaćenih, broj preostalih na čekanju i prihod.**
- Ulazi su nenegativni cijeli brojevi.
- Dodatni test: dostupni=2, čekaju=5 → 2 3 3.

Model rješenja: prototipovi i integracija2

Funkcija prima vrijednosti argumenata, ne pristupa lokalnim varijablama funkcije main().

```
int primiBicikle(int dostupni, int cekaju);

double cijenaPoBiciklu(int prihvaceni);

double izracunajPrihod(int prihvaceni, double cijena);

int main() {

    int prihvaceni = primiBicikle(6, 5);

    double cijena = cijenaPoBiciklu(prihvaceni);

    double prihod = izracunajPrihod(prihvaceni, cijena);

    cout << prihvaceni << " " << (5 - prihvaceni) << " " << prihod;

}
```

```
int primiBicikle(int dostupni, int cekaju) {
    return (dostupni < cekaju) ? dostupni : cekaju;
}

double cijenaPoBiciklu(int prihvaceni) {
    return (prihvaceni >= 4) ? 1.20 : 1.50;
}

double izracunajPrihod(int prihvaceni, double cijena) {
    return prihvaceni * cijena;
}
```

Vođena rasprava i tipične pogreške

- Zašto cijenaPoBiciklu prima prihvaćeni, a ne ukupni broj onih koji čekaju?

Jer popust ovisi o stvarno posluženim biciklima.

- Je li parametar prihvaceni u cijenaPoBiciklu ista varijabla kao u main?

Nije: vrijednost se prenosi, doseg je zaseban.

- Potpora: unaprijed zadana tablica „funkcija – ulaz – izlaz – odgovornost” i djelomičan dijagram.
- napisati osobne testove prije izvođenja i objasniti zašto je dizajn funkcija prikladan.
- Al smije kritički provjeravati rubne slučajeve **nakon osobnog pokušaja**.
- Formativno vrednovanje: procjenjuje se argumentacija izbora funkcija, praćenje podataka i ispravna predviđanja;