



Programiranje – SIU1- funkcije i rekurzija u programiranju

zašto program rastavljamo na funkcionalne cjeline, zašto obrazac **ulaz** → **obrada** → **izlaz**

Funkcije kao funkcionalne cjeline programa

- *Odrediti ulazne i izlazne parametre funkcionalnih cjelina.*

Ishodi nastavnog sata - Na kraju sata učenik će moći:

- objasniti zašto složeniji problem dijelimo na manje funkcionalne cjeline;
- odrediti ulazne podatke i očekivani rezultat jednostavnog problema;
- prepoznati postupak koji se može izdvojiti u zasebnu funkcionalnu cjelinu;
- opisati funkcionalnu cjelinu modelom **ulaz** → **obrada** → **izlaz**;
- predložiti naziv funkcije koji opisuje njezinu namjenu;
- razlikovati problem koji funkcija rješava od podataka koje prima i rezultata koji daje;
- obrazložiti zašto izdvajanje funkcionalnih cjelina može olakšati izradu i provjeru programa.

Kako složeniji problem organizirati tako da ga možemo riješiti, provjeriti i objasniti?

- prvo razmišljanje bez kôda,
- traženje obrasca,
- rješavanje malog slučaja,
- oblikovanje algoritma,
- implementacija i testiranje rubnih slučajeva.

Početni izazov – „Prolazi li?“

- Učenik na natjecanju rješava tri zadatka i iz svakoga može osvojiti od 0 do 100 bodova.

U sljedeći krug prolazi ako ima **najmanje 200 bodova ukupno i najmanje 40 bodova iz svakog zadatka.**

- Za bodove **75, 82, 51** odredi prolazi li učenik dalje. Zapiši postupak.
- A ako su bodovi **100, 100, 35**? učenik ipak ne prolazi.
- Prvih nekoliko minuta **nema programiranja**. što znamo, što trebamo dobiti, koji su uvjeti?

- **ULAZ:** tri broja bodova

↓

OBRADA: zbroj + provjera pojedinačnih bodova

↓

IZLAZ: prolazi / ne prolazi

- Kad bismo veliki program podijelili na manje dijelove, možemo li ovaj postupak izdvojiti kao jednu cjelinu?
- *Koji dio možemo pretvoriti u zasebnu funkcionalnu cjelinu?*
- što funkcija mora znati i što ostatak programa od nje treba dobiti
- Kako bismo je nazvali? prolazi, provjeri, prolaz ...
- `bool prolazi(int a, int b, int c);`

Mini izazov – „Projektiraj funkciju“

FUNKCIJA	ULAZ	IZLAZ
----------	------	-------

prvo samostalno razmišljanje, zatim usporedba s partnerom

U bilježnici za svaki problem:

Problem A: Za zadani cijeli broj treba odrediti je li paran.

Problem B: Za tri duljine treba odrediti mogu li predstavljati stranice trokuta.

Problem C: „Je li broj prost?“

Problem D: „Koliko je znamenaka broja?“

Problem E: „Koji je najveći od tri broja?“

Mora li funkcija vratiti rezultat izračuna ili samo odgovor na pitanje?

Razlika: **ulazni podatci _ rezultat koji je potreban ostatku programa.**

Problem C

- Natjecatelj je osvojio određeni broj bodova. Ako je rezultat veći od 100, zapisuje se 100. Ako je manji od 0, zapisuje se 0. U ostalim slučajevima ostaje nepromijenjen. Osmislite funkcionalnu cjelinu koja određuje konačni broj bodova.
- **Zašto funkcija mora dobiti baš te podatke?**
Treba li joj još nešto?
Što ostatak programa od nje očekuje?

- Trebamo funkciju koja određuje je li broj djeljiv sa 7.

- `int provjeri(int broj, int sedam, int rezultat);`

- **Što vam se u ovom prijedlogu ne sviđa?**

- Treba li funkcija primiti sedam?
- Je li **provjeri** dovoljno dobar naziv?

ULAZ → OBRADA → IZLAZ

- **FUNKCIONALNA CJELINA / FUNKCIJA**

Problem dijelimo na manje cjeline koje imaju određenu zadaću.

- Kada osmišljam funkciju pitam:
- Što funkcija treba napraviti?
- Koje podatke za to mora dobiti?
- Što nakon obrade treba dati ostatku programa?
- Kako je mogu smisleno nazvati?
- što programer stvarno radi prije implementacije: interpretira zahtjev, traži postupak, određuje podatke, oblikuje funkcionalnu cjelinu, razmatra alternativu i testira ideju.

PROBLEM →
FUNKCIONALNA
CJELINA → ULAZ →
OBRADA → IZLAZ

u problemu **prepoznati što jedan dio
programa treba dobiti i što treba
proizvesti**



2. sat

SIU1- funkcije i rekurzija u programiranju
Od funkcionalne cjeline do C++ funkcije

Od funkcionalne cjeline do C++ funkcije

Ishod SIU-a kojem sat prvenstveno doprinosi: *Odrediti ulazne i izlazne parametre funkcionalnih cjelina.*

- **poziv nasuprot definiciji i prednosti funkcija u odnosu na ponavljanje kôda.**

Ishodi nastavnog sata

- Na kraju sata učenik će moći:
- na temelju problemskog zahtjeva odrediti ulazne podatke i rezultat funkcije;
- razlikovati formalne parametre funkcije od stvarnih argumenata pri njezinu pozivu;
- razlikovati definiciju funkcije od njezina poziva;
- povezati model **ulaz** → **obrada** → **izlaz** s parametrima, tijelom funkcije i povratnom vrijednošću;
- zapisati jednostavnu funkciju i njezin poziv;
- predvidjeti rezultat jednostavnog programa koji koristi funkciju;
- obrazložiti zašto je u određenoj situaciji izdvajanje postupka u funkciju bolje od ponavljanja istoga programskog kôda.

„što funkcija treba znati?“

- Na natjecanju se konačni broj bodova dobiva tako da se osnovnim bodovima pribroji bonus, ali rezultat ne može biti veći od 100.
Što bi funkcija koja određuje konačni rezultat trebala **primiti**, a što **vratiti**?
- U bilježnicu:
- **ULAZ:** ...
IZLAZ: ...
- usporedba
- **ULAZ → OBRADA → IZLAZ**

Od ideje do C++ funkcije

```
int konacniBodovi(int bodovi, int bonus)
{
    int rezultat = bodovi + bonus;

    if (rezultat > 100)
        rezultat = 100;

    return rezultat;
}
```

ULAZ → int bodovi, int bonus

OBRADA → zbrajanje i provjera

IZLAZ → return rezultat

Što funkcija mora znati prije nego počne raditi?

Koji dio predstavlja obradu?

Što ostatak programa dobiva nakon završetka funkcije?

Zašto funkciji ne predajemo i rezultat?

Pojmovi:

naziv funkcije,
formalni parametri,
tijelo funkcije,
povratna vrijednost.

Definicija nije isto što i poziv

```
int konacniBodovi(int bodovi, int bonus)
{
    int rezultat = bodovi + bonus;

    if (rezultat > 100)
        rezultat = 100;

    return rezultat;
}

int main()
{
    cout << konacniBodovi(70, 15) << endl;
    cout << konacniBodovi(95, 20) << endl;
}
```

Gdje je funkcija definirana?

Koliko je puta pozvana?

Koji su formalni parametri?

Koji su stvarni argumenti prvog poziva?

Što će program ispisati?

Poziv	bodovi	bonus	rezultat
konacniBodovi(70,15)	70	15	?
konacniBodovi(95,20)	95	20	?
konacniBodovi(40,25)	?	?	?

„Zašto uopće funkcija?”

```
int a = 95, bonusA = 20;  
int rezultatA = a + bonusA;  
if (rezultatA > 100)  
    rezultatA = 100;
```

```
int b = 63, bonusB = 12;  
int rezultatB = b + bonusB;  
if (rezultatB > 100)  
    rezultatB = 100;
```

```
int c = 88, bonusC = 17;  
int rezultatC = c + bonusC;  
if (rezultatC > 100)  
    rezultatC = 100;
```

„Što vam se ne sviđa u ovom programu?”

Organizator mijenja pravilo: najveći rezultat više nije 100 nego 120. Koliko mjesta moramo mijenjati?

A ako postupak imamo u jednoj funkciji?

razlozi za uporabu funkcija.

funkcija omogućuje:

- izbjegavanje nepotrebnog ponavljanja;
- ponovno korištenje postupka;
- lakše mijenjanje programa;
- lakše testiranje pojedinog dijela;
- pregledniju strukturu programa.

na konkretnom programu objasniti prednost.

Mini izazov – „Ulaznica za finale“

- **Ulaznica za finale**
- Natjecatelj je riješio određeni broj zadataka i ostvario određeni broj bodova.
U finale prolazi ako je riješio **najmanje 3 zadatka** i osvojio **najmanje 60 bodova**.
- Osmisli funkciju koja određuje prolazi li natjecatelj u finale.

Faza A – 2 minute individualno zapisati:

- **naziv funkcije:**
ulaz:
izlaz:

Faza B – 3 minute u paru

- Usporediti rješenja i pokušati zapisati zaglavlje funkcije.

nastavak

- `bool prolazi(int brojZadataka, int bodovi);`

- **Faza C – 3–4 minute**

- Pokušati na papiru dopuniti funkciju

- **Faza D – „Natjecateljski“ dio**

- **Koja biste tri testna slučaja odabrali da provjerite je li rješenje ispravno?**

- Vrlo su zanimljive upravo granice:

- 3, 60
2, 100
5, 59

```
bool prolazi(int brojZadataka, int bodovi)
{
    return brojZadataka >= 3 && bodovi >= 60;
}
```

Je li 5, 80 dovoljno dobar test? Što njime zapravo provjeravamo?

- navika **traženja testova koji mogu otkriti pogrešku.**

definicija/poziv funkcije, parametri i povratna vrijednost

PROBLEM → FUNKCIJA → DEFINICIJA
→ POZIV → REZULTAT

formalni parametri ← **konkretni
argumenti pri pozivu**

„Gdje opisujemo što funkcija radi?“ —
u definiciji;

„Kada se to stvarno izvršava?“ — *kad
funkciju pozovemo.*



3. sat

SIU1- funkcije i rekurzija u programiranju

**Odrediti ulazne i izlazne parametre
funkcionalnih cjelina**

Projektiranje funkcije prema problemu

Ishodi nastavnog sata

Na kraju sata učenik će moći:

- analizirati novi problemski zahtjev i izdvojiti podatke potrebne za rješavanje;
- prepoznati postupak koji je smisleno izdvojiti u funkciju;
- samostalno odrediti ulazne parametre i povratnu vrijednost funkcije;
- oblikovati odgovarajući potpis funkcije;
- predvidjeti rezultat poziva funkcije za zadane argumente;
- procijeniti je li predloženo sučelje funkcije primjereno problemu;
- obrazložiti vlastiti izbor parametara i rezultata funkcije.

Aktivacija – koja je funkcija bolja?

- Funkcija treba odrediti je li učenik položio ispit. Za prolaz je potrebno najmanje 50 bodova.

A. `bool polozen(int bodovi);`

B. `bool polozen(int bodovi, int prag, bool rezultat);`

- Bez razgovora u bilježnicu: **A ili B? Zašto?**
- Nakon minute-dvije razgovora u paru: **Treba li funkcija dobivati podatak koji joj nije potreban ili podatak koji može sama odrediti?** dobro sučelje funkcije ovisi o zahtjevu problema
- Prepoznati prvi potpis kao prikladniji jer je prag zadan pravilom problema, a rezultat funkcija treba proizvesti.
- „A što ako različiti ispiti imaju različit prag prolaznosti?“ `bool polozen(int bodovi, int prag);`

Problem 1 – Popust

- Internetska trgovina kupcu daje 10 % popusta ako vrijednost kupnje iznosi najmanje 100 €. Potrebno je odrediti konačnu cijenu koju kupac plaća.

Faza 1 – individualno, 2–3 minute

- Što znamo?
Što trebamo dobiti?
Što funkcija treba primiti?
Što treba vratiti?
Kako bismo je nazvali?

- `double konacnaCijena(double cijena);`

Faza 2 – par, 2–3 minute

- Usporedba rješenja i funkcija na papiru.

```
double konacnaCijena(double cijena)
{
    if (cijena >= 100)
        return cijena * 0.9;

    return cijena;
}
```

Koje biste vrijednosti testirali?

Koji je opasniji test: 150 ili 100? Zašto?

99, 100, 101.

razvijamo razmišljanje o **graničnim slučajevima**.

„Pokvari funkciju“ – analiza loših prijedloga

- `double popust(double cijena);`
- `double popust(double cijena, double konacna);`
- `bool popust(double cijena);`
- **Koji biste prijedlog prihvatili, koji odbacili, a za koji trebate dodatno objašnjenje autora?**
- Primjerice, naziv `popust` nije idealan ako funkcija zapravo vraća konačnu cijenu.
- `bool` nije odgovarajući ako trebamo dobiti iznos.
- `konacna` kao ulazni parametar nema smisla ako ga funkcija upravo izračunava.
- **Dobra funkcija ne dobiva sve podatke koje program ima. Dobiva podatke koji su potrebni za njezinu zadaću.**

Algoritamski izazov – Parking

Parkiralište naplaćuje parkiranje prema trajanju:

- do 1 sata: 2 €
- više od 1 do uključivo 3 sata: 5 €
- više od 3 sata: 8 €

Za zadani broj sati parkiranja treba odrediti cijenu.

A – samostalno,

- Bez razgovora:
- **ULAZ:** _____
IZLAZ: _____
NAZIV FUNKCIJE: _____
- i pokušaj potpisa funkcije.
- `double cijenaParkiranja(double sati);`

B – rad u paru,

Usporedba prijedloga.

odrediti očekivani rezultat za:

0.5 → 2 €

1 → 2 €

2 → 5 €

3 → 5 €

4 → 8 €

Koji su od ovih testova posebno važni?

1 i 3 trebali bi se izdvojiti kao granice.

C – napiši funkciju na papiru, samostalno.

```
double cijenaParkiranja(double sati)

{
    if (sati <= 1)
        return 2;

    if (sati <= 3)
        return 5;

    return 8;
}
```

D – „skriveni test“,

Zamislite da ste autor programerskog natjecanja. Smijete odabrati samo jedan skriveni test kojim želite otkriti što više pogrešnih rješenja. Koji biste odabrali?

kako bi rješenje moglo biti pogrešno?

Sistematizacija – četiri pitanja prije pisanja funkcije

- 1. Što funkcija treba napraviti?**
- 2. Koje podatke mora dobiti da bi to mogla napraviti?**
- 3. Što treba vratiti ostatku programa?**
- 4. Kako ću provjeriti radi li ispravno?**