Programiranje – SIU1 – i4 – 1. sat

Dekompozicija - Identificirati i izdvojiti manje cjeline u problemu te ih integrirati u rješenje.

- prepoznavanje smislenih manjih cjelina,
- određivanje njihove odgovornosti te njihovo povezivanje u rješenje.
- **PROBLEM -> MANJE CJELINE -> ODGOVORNOST FUNKCIJE -> ULAZI/IZLAZ -> POVEZIVANJE -> TEST CJELINE -> TEST RJEŠENJA**

- **Koji se dio problema može riješiti kao zasebna cjelina?**
- **Što je točno odgovornost te funkcije?**
- **Koje podatke prima i što vraća?**

- **Tko koristi rezultat te funkcije i kako se cjeline povezuju?**

Od velikog problema do manjih odgovornosti - ishodi

- prepoznati dijelove problema koji imaju zasebnu, jasno opisivu zadaću;
- predložiti smislene funkcionalne cjeline i njihove nazive;
- odrediti ulazne parametre i povratnu vrijednost pojedine funkcije;
- razlikovati korisnu dekompoziciju od nepotrebnog usitnjavanja;
- objasniti zašto je rješenje s dobro podijeljenim odgovornostima lakše provjeriti i mijenjati.

- Glavna ideja: dobra podjela nije „što više funkcija“, nego **odgovarajući broj funkcija** od kojih **svaka ima jednu jasnu odgovornost**.

Problemski ulaz: ŠKOLSKI TURNIR

- Za svakog natjecatelja poznata su tri rezultata. Program treba izračunati ukupan broj bodova, odrediti je li natjecatelj prošao dalje (najmanje 200 ukupno i najmanje 40 u svakom zadatku) te na kraju ispisati kratku poruku s ukupnim rezultatom i odlukom.

- Može li se ova zadaća opisati jednom kratkom rečenicom?

- Jedna funkcija - jedna jasna odgovornost. **Izbjegavati funkcije koje „rade sve“.**

- prije bilo kakvog koda: Što su ovdje **različite odgovornosti/zaduženja/zadaci**?

- usporediti popis koraka i pokušati ih grupirati u 3-4 smislene cjeline.
- Za svaku cjelinu odrediti naziv, ulaze i izlaz

- izračunati ukupan broj bodova;
- provjeriti uvjet prolaska;
- prikazati rezultat korisniku.

- Moguća struktura - nakon rasprave:
- int ukupno(int a, int b, int c);
  bool prolazi(int a, int b, int c);
  void ispisiRezultat(int zbroj, bool prolaz);
- Važno: ovo nije jedino ispravno rješenje uz uvjet da je moguće objasniti odgovornosti i tok podataka.

Loša dekompozicija

- A) Jedna funkcija `void obradiSve(...)` računa zbroj, provjerava prolaz i ispisuje poruku.
- B) Posebne funkcije `zbrojiPrvaDva()`, `dodajTreci()`, `provjeriPrvi()`, `provjeriDrugi()`, `provjeriTreci()`, `ispisiDA()`, `ispisiNE()`.

- Pitanja:
- Što je problem u A, a što u B?
- Analizirati dva ponuđena rješenja: premalo i previše funkcija.
- Kada je izdvajanje funkcije korisno, a kada samo usitnjava program?

Mini izazov: DOSTAVA DRONOM

- Dron treba dostaviti paket. Poznate su udaljenost do odredišta, masa paketa i trenutačna razina baterije. Program treba izračunati procijenjenu potrošnju baterije, provjeriti ima li dovoljno baterije uz sigurnosnu rezervu te prikazati može li dron krenuti.
- Bez pisanja programa predložiti 2-3 funkcije.
- Za svaku zapisati: NAZIV | ULAZI | IZLAZ | ODGOVORNOST/ZADATAK.

- **broj funkcija nije cilj; cilj je smisleno razdvajanje odgovornosti**.
- Samostalno predložiti funkcionalne cjeline i usporediti rješenja.
- Navesti strukturu rješenja i argumentaciju, zapisati zajednički obrazac dekompozicije.
- Povezati s dosadašnjim znanjem o parametrima i povratnoj vrijednosti.

- Dekompozicija = podjela problema na manje smislene cjeline.
- Funkcija treba imati jasnu odgovornost.
- Za svaku funkciju pitam: što prima, što radi, što vraća?
- Dobra podjela olakšava testiranje, razumijevanje i promjenu programa.

Programiranje – SIU1 – i4 – 2. sat

Dekompozicija - Identificirati i izdvojiti manje cjeline u problemu te ih integrirati u rješenje – kako funkcije surađuju.

izdvojene funkcije postaju rješenje tek kada jasno odredimo tok podataka i povežemo ih u cjelinu – ishodi:

- povezati više funkcija pozivima iz glavnog programa;
- pratiti kako rezultat jedne funkcije postaje ulaz drugoj funkciji;
- predvidjeti redoslijed poziva i međurezultate bez pokretanja programa;
- uočiti problem u loše povezanom rješenju;
- osmisliti strukturu rješenja novog problema s više funkcionalnih cjelina.

Aktivacija: od funkcija do programa, tok podataka (TURNIR)

- Promotriti strukturu:
- int z = ukupno(75, 82, 51);
  bool p = prolazi(75, 82, 51);
  ispisiRezultat(z, p);

Pratiti kod i crtati strelice između rezultata i sljedećih poziva.	**Rezultat funkcije treba završiti - usmjeriti tamo gdje je potreban.**

- Koja se funkcija poziva prva?
- Koje dvije vrijednosti treba imati prije poziva `ispisiRezultat`?
- Bi li se redoslijed prvih dviju funkcija mogao zamijeniti? Zašto?
- Što bi se promijenilo kada bi `prolazi` kao parametar primala i već izračunati zbroj?

Pokvareni program - pronađi strukturni problem

- double cijenaSDostavom(double cijena, double dostava) { return cijena + dostava; }
  bool besplatnaDostava(double cijena) { return cijena >= 50; }

  // main
  double ukupno = cijenaSDostavom(48, 5);
  bool besplatno = besplatnaDostava(ukupno);

U paru otkriti gdje je narušen tok podataka ili odgovornost funkcije.	**analizirati strukturu.**

- Pravilo trgovine glasi: **besplatna dostava ovisi o vrijednosti KOŠARICE prije dodavanja dostave.**
- Je li program strukturno ispravno povezao funkcije?
- objasniti pogrešku i predložiti tok podataka.
- Očekivana ideja: funkcija `besplatnaDostava` mora dobiti izvornu cijenu košarice, a ne već uvećanu ukupnu cijenu. Ovo je pogreška u povezivanju cjelina, ne u samim funkcijama.

Glavni problem: PAMETNA DOSTAVA

- Mala dostavna služba određuje cijenu i mogućnost isporuke. Za narudžbu su **poznati vrijednost košarice, udaljenost u kilometrima i masa paketa**. Dostava je moguća samo **ako masa nije veća od 20 kg i udaljenost nije veća od 30 km**. **Osnovna cijena dostave iznosi 3 eura, a dodaje se 0,40 eura po kilometru**. **Ako je vrijednost košarice najmanje 80 eura, kupac dobiva popust od 3 eura na izračunatu cijenu dostave**, ali cijena dostave ne može biti negativna. Program na kraju treba prikazati **je li dostava moguća i, ako jest, konačnu cijenu dostave**.
- Zadatak:

1. Izdvojiti 2-4 smislene funkcionalne cjeline.
2. Za svaku zapisati odgovornost, ulazne parametre i povratnu vrijednost.
3. Nacrtati ili zapisati redoslijed poziva i tok podataka između funkcija.
4. Predvidjeti rezultat za dva zadana testa.
5. Smisliti jedan rubni test koji provjerava točno jednu važnu granicu.

Zadani testovi

Košarica	Udaljenost	Masa	Očekivanje
50 €	10 km	5 kg	dostava moguća; cijena 7 €
100 €	35 km	4 kg	dostava nije moguća
80 €	5 km	20 kg	dostava moguća; cijena 2 €

Samostalno razložiti problem, definirati funkcije i složiti redoslijed poziva.	**obrazloženje zašto je baš ta podjela smislena.**

Jedna moguća, ali ne jedina struktura:

bool mogucaDostava(double udaljenost, double masa);
double osnovnaDostava(double udaljenost);
double primijeniPopust(double cijenaDostave, double vrijednostKosarice);

Prihvatiti dekompoziciju ako su odgovornosti jasne i cjeline se mogu smisleno povezati.

Pitanja za usporedbu rješenja

- Je li bolje imati jednu funkciju `cijenaDostave` koja u sebi rješava i popust ili dvije funkcije? Oba odgovora mogu biti obranjiva - što dobivamo, a što gubimo?
- Treba li funkcija koja računa cijenu znati masu paketa ako masa utječe samo na mogućnost dostave?
- Koju funkciju možemo testirati potpuno neovisno o ostalima?
- Gdje bi se najlakše promijenilo pravilo „80 eura“ ako trgovina promijeni prag?

Usporediti različite valjane dekompozicije.	**Ne postoji nužno samo jedna dobra struktura.**

- Integracija = povezivanje funkcionalnih cjelina u cjelovito rješenje.
- Rezultat jedne funkcije može postati ulaz drugoj funkciji ili se koristiti u glavnom programu.
- Funkcija ne treba primati podatke koji joj nisu potrebni za njezinu odgovornost.
- Dobro rješenje testira pojedine funkcije i zatim njihovu suradnju.

Programiranje – SIU1 – i4 – 3. sat

Dekompozicija - Identificirati i izdvojiti manje cjeline u problemu te ih integrirati u rješenje – lokalne i globalne varijable.

Ishodi:
Povezivanje funkcionalnih cjelina: tijek podataka i doseg varijabli

- kako se podaci prenose između funkcija i glavnog programa,
- razlika lokalnih i globalnih varijabli
- zašto isto ime varijable u različitim funkcijama ne znači nužno istu varijablu?
- Primjena pri povezivanju više funkcija u jednostavno rješenje.
- Skica tijeka: main → funkcija A → rezultat → funkcija B → ispis.
- Točno predvidjeti vrijednosti lokalnih varijabli pri pozivu funkcije i objasniti njihov doseg.
- Prepoznati zasjenjivanje (shadowing) i razlikovati lokalnu varijablu od globalne.
- Predložiti integraciju najmanje dviju funkcija uz prijenos podataka parametrima i povratnim vrijednostima, bez nepotrebnih globalnih varijabli.

Problemski uvod

- „Dva razredna odjela škole koriste varijablu trenutni_broj_ucenika_u_razredu. Hoće li promjena u jednome promijeniti drugu?”

Program A – parametar nije ista varijabla

- ```
  int povecaj(int broj) {
      broj += 3;
      return broj;
  }
  int main() {
      int broj = 5;
      int novi = povecaj(broj);
      cout << broj << " " << novi;
  }
  ```

Pitanja:

- Što će se ispisati?
- Koji je doseg varijable broj u funkciji, a koji u main?
- Zašto poziv ne mijenja broj u main? Očekivano: 5 8; dva različita lokalna objekta; argument se ovdje prenosi po vrijednosti.
  ```

Program B – globalna varijabla i zasjenjivanje

- int bodovi = 10; // globalna varijabla
  int dodaj(int x) {
    int bodovi = x + 2; // lokalna zasjenjuje globalnu
    return bodovi;
  }
  int main() {
    int rezultat = dodaj(5);
    cout << rezultat << " " << bodovi;
  }

Koji bodovi koristi return?
Zašto se globalni bodovi nisu promijenili?
Je li globalna varijabla ovdje potrebna?

doseg označuje gdje je ime dostupno;

životni vijek varijable nije isto što i doseg.

Globalne varijable nisu zabranjene, ali za ovaj problem nepotrebno otežavaju razumijevanje i testiranje.

- što se ispisuje i zašto? (Očekivani ispis: 7 10. lokalna bodovi zasjenjuje globalnu samo unutar funkcije.)

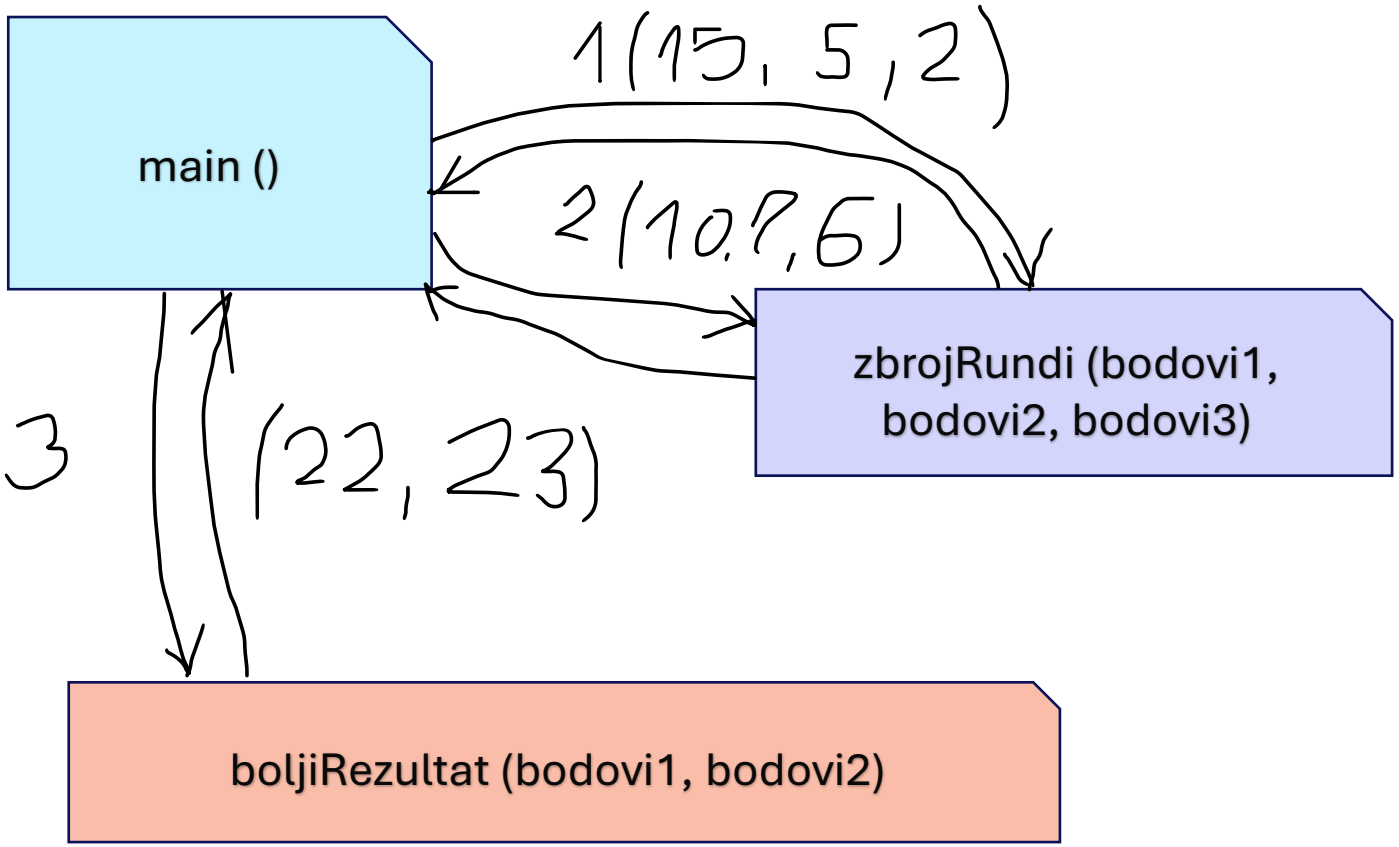
Problemski rad u paru – Školski turnir

Za dva natjecatelja poznati su bodovi iz tri runde za svakog. Funkcija zbrojRundi(a,b,c) vraća ukupne bodove. Funkcija boljiRezultat(prvi,drugi) vraća veći od dvaju ukupnih rezultata.

U main () se učitava šest brojeva, pozivaju funkcije i ispisuje pobjednički rezultat. Bez grananja u main (); odluka je odgovornost druge funkcije.

- Nacrtaj strelice prijenosa podataka i napiši potpise dviju funkcija.
- Odredi koje su varijable lokalne funkciji main (), a koje pojedinoj funkciji.
- Napiši 3–5 naredbi u main () koje povezuju rezultate (pretpostavi da su ulazne vrijednosti već učitane).
- Dodatno: može li se zadatak riješiti bez globalnih varijabli? Objasni.

- Primjer integracije: int prvi = zbrojRundi(a1,a2,a3); int drugi = zbrojRundi(b1,b2,b3); int najbolji = boljiRezultat(prvi,drugi); cout << najbolji; Funkcija boljiRezultat može koristiti if ili uvjetni operator;



- Vidi li funkcija zbrojRundi() varijablu ukupnoBodovi iz main()? Ne. Ona ima vlastite lokalne parametre
- Kako funkcija BoljiRezultat() dobiva broj 22? Kao vrijednost argumenta koji se prosljeđuje njezinu parametru bodovi.
- Jesu li bodovi1 ista varijabla? Nisu. Pripadaju različitim funkcijama.

- Ako funkcija prima int x po vrijednosti i izvrši x++, mijenja li se istoimena varijabla u main? (Ne; ona je kopija vrijednosti.)
- Kako rezultat funkcije zbrojRundi postaje ulaz funkciji boljiRezultat? (Povratnom vrijednošću koja se sprema ili izravno prosljeđuje kao argument.)

Završno

Jasno razlikovati lokalni doseg, globalni doseg, zasjenjivanje i prijenos po vrijednosti.

Ne poistovjećivati doseg i životni vijek.

varijable deklarirane u funkciji nisu automatski dostupne drugim funkcijama; informacije se prenose parametrima i povratnim vrijednostima.