Što je doseg varijable?

- područje programa u kojem je varijabla vidljiva i u kojem se može koristiti

Da li vrijeme života varijable treba minimizirati ili maksimizirati? Zašto?

- Vrijeme života = broj naredbi kroz koje varijabla živi, počinje u prvoj naredbi u kojoj se koristi, a završava u zadnjoj naredbi u kojoj se koristi
- Potrebno ga je minimizirati jer :
  - smanjuje se prostor ranjivosti varijable
  - daje jasniji i čitljiviji kod
  - smanjuje se vjerojatnost pogrešaka uslijed pogrešne inicijalizacije varijable
  - lakše je podijeliti postupke u više manjih

Što je to vrijeme povezivanja varijable?

- Trenutak u kojem se varijabla povezuje sa svojoj vrijednosti
- vrste vremena povezivanja:
  - u trenu pisanja programa (direktno upisivanje vrijednosti)
  - u trenu prevođenja programa (const float AVERAGE = 3.1; pa onda koristis varijablu za razliku od prethodnog gdje uvijek pišeš 3.1)
  - u trenu izvršavanja programa (čitanje vrijednosti varijable iz datoteke)

Nabrojati smjernice za pravilno imenovanje varijabli

- ime varijable mora u potpunosti i točno opisivati entitet kojeg varijabla predstavlja
- varijablu imenovati riječima koje je opisuju
- konvencija za imenovanje varijabli u C#-u:

Vrsta varijable	Konvencija imenovanja
cjelobrojni indeksi	`i, j`
klase i tipovi	`VelikaMalaSlova`
privatne varijable i postupci	`imePrivatneVarijablePostupka`
javne varijable i postupci	`ImeJavneVarijablePostupka`
konstante	`SVA_VELIKA_SLOVA`

Koje su prednosti korištenja konstanti?

- Parametiziranje programa
- promjene na jednom mjestu reflektiraju se na sva mjesta gdje se konstanta koristi
- dobre za definiciju veličine nekog podatka

Kako uspoređivati decimalne brojeve?

- Ne direktno! (if totalSum == REAL_NUMBER)
- uzeti u obzir grešku pri zaokruživanju (npr. ERROR = 0.0001; if abs(totalSum – REAL_NUBMER) < ERROR)

Što su to magični brojevi?

- Brojevi koji se pojavljuju direktno u kodu
- zamijeniti ih imenovanim konstantama ili pretprocesorskim direktivama
- jedino se 0 i 1 smiju pojavljivati u kodu

Nabrojati smjernice za pravilno korištenje if naredbe

- prvo napisati glavni (nominalni) slučaj, a tek zatim manje česte slučajeve
- provjeriti jesu li operatori usporedbe dobro upotrebljeni
- Napisati glavni slučaj uz if, radije nego uz else

Koje su karakteristike switch-case naredbe?

- Razlikuje se od programskog jezika do programskog jezika (npr. C#, Java – case uzima po jednu vrijednost, Visual Basic – može poprimiti i niz vrijednosti)
- ako ima puno case-ova moguće je: poredati ih abecedno ili numerički uzlazno, staviti glavni slučaj prvi, poredati ih po frekvenciji korištenja
- izbjegavati propadanje kroz case grane jer ugnježđuje kontrolne naredbe (wtf?)

Što je to pojava pomiješanih indeksa?

- Korištenje jedno te istog indeksa u dvije namjene – bolje koristiti smislenija imena

Koje vrste kohezije postoje? (165)

- Najpoželjnija
  - Funkcionalna – javlja se kada postupak obavlja samo jedan zadatak
- Manje poželjne
  - Sekvencijalna – kada postupak sadrži zadatke koje se s razlogom moraju izvršiti u točno određenom redoslijedu
  - Komunikacijska – kada zadaci unutar postupka nisu ni na koji drugi način povezani osim što koriste iste podatke
  - Vremenska – kada se zadaci ujedinjuju u postupak samo zato jer se izvršavaju u isto vrijeme
- Nepoželjne
  - Proceduralna – kada postupak sadrži zadatke koji se BEZ RAZLOGA moraju izvršiti u točno određenom redoslijedu
  - Logička – kada se nekoliko zadataka ugura u jedan te isti postupak te se s parametrom postupka odabire koji se od njih izvršava
  - Slučajna – kada zadaci u postupku nemaju ništa zajedničko

Nabrojati osnovne smjernice za pisanje postupaka

- imati samo funkcionalnu koheziju
- ime postupka mora opisivati točno ono što postupak radi i ne biti preopćenita
- u ime postupka uključiti glagol, najbolje da se sastoji od glagola (signalizita radnju) i imena objekta (nad cime se radnja odvija)
- ne pisati preduge postupke (ni slučajno više od 200 linija koda)

Nabrojati osnovne smjernice za korištenje parametara postupaka

- postaviti parametre u redoslijedu: 1. ulazni – 2. oni koji se modificiraju – 3. izlazni
- u svim postupcima koji koriste slične parametre poredati parametre na isti način
- postupak mora koristiti sve svoje parametre
- statusne varijable koristiti kao zadnje parametre
- ograničiti broj parametara postupka na otprilike 7
- ne koristiti parametre kao varijable za izračun unutar postupka tj. Potruditi se da budu konstanti
- koristiti imenovane parametre u prog jezicima koji ih podržavaju

Nabrojati smjernice za pisanje kompleksnih logičkih izraza

- potrebno je prelomiti u više programskih linija, ali da to bude uočljivo

Koja je razlika između razreda i objekta?

- Klasa (razred) opisuje cjeline koje imaju nešto (atribute i ponašanje) zajedničko, a objekt je pojedina instanca razreda.

Osnovni koncepti OOP-a

- apstrakcija
  - modeliranje objekata na način da se koriste samo bitne komponente stvarnog objekta
- nasljeđivanje
  - odnos između klasa kod kojeg se jedna klasa stvara na temelju druge tako da joj se

dodaju specifični atributi i ponašanje
- višeobličje (polimorfizam)
  - slična ponašanja klasa, svaka klasa za sebe precizno definira zajedničko ponašanje
- učahurivanje (enkapsulacija)
  - klasa skriva neke atribute i ponašanja od ostalih klasa – to se radi korištenjem modifikatora vidljivosti
- slanje poruka
- asocijacija (povezanost)
  - povezanost objekata (npr. Daljinski i tv), dvosmjerna – ako oba objekta mogu slati poruke jedno drugome, jednosmjerna – ako samo jedan može
- agregacija (sadržavanje)
  - objekt sastavljen od više drugih objekata (npr. Računalo sastavljeno od CPU, monitora, miša...)

Nabrojati principe objektno orijentiranog dizajna
- Minimizirati dostupnost članskih varijabli i metoda klase
- Češće koristit sadržavanje, manje nasljeđivanje
- Programirati u odnosu na sučelje, ne implementaciju
- Open/Closed principle
- Liskov substitution principle
- Dependecy Inversion principle
- Interface segragation principle

Koje su značajke lošeg objektno orijentiranog dizajna?
- Rigidnost – teško raditi izmjene jer utječu na puno drugih dijelova sustava
- Krhkost – nakon izmjene na jednom dijelu dolazi do neočekivanih kvarova na drugom
- Slaba mobilnost – nije moguće izdvojiti dio sustava

Koje su četiri osnovne karakteristike kojima se opisuju oblikovni obrasci?
- Naziv oblikovnog obrasca
- Problem
- Rješenje
- Posljedice

Aspektno orijentirano programiranje je programska paradigma koja povećava modularnost oddatnim razdvajanjem poslova unutar programa