**USMENI PPiJ**

**- Što je beta-redukcija?**

Beta-redukcija je proces izračunavanja rezultata primjenom funkcije na izrazu.

**- karakteristike F#**

Nepromjenjivost, višedretvenost, paralelizam, asinkronost, nepotrebnost definiranja tipova

**- vezano za indexe u petlji (index cross talk)**

Pomiješani indeksi (index cross-talk) -> ugniježđene petlje - česta pogreška je zamjena indeksa gdje Array[i] postaje Array[j] i obrnuto.

**- Što sve sadržava apstraktna metoda?**

Modifikator pristupa (public, private…), povratni tip (void, int…), naziv metode i argumente koje metoda prima. Dakle sve što i “normalna” metoda, ali bez implementacije (tijela).

**- Što su virtualne klase, a što virtualne metode? (TRIK PITANJE! Virtualne klase ne postoje!)**

**→ postoje u c++ ali ako to radis nekaj radis krivo :D http://stackoverflow.com/questions/21558/in-c-what-is-a-virtual-base-class**

Virtualna metoda, za razliku od apstraktne metode, nudi implementaciju. Ako korisniku ta implementacija ne odgovara on ju može nadjačati.

**- Nazivlje varijabli u c i c++ ?**

Varijable se moraju inicijalizirati prije korištenja u kodu, mogu se sastojati od engleskih velikih i malih slova, podvlake (\_), i brojeva, moraju imati definiran tip varijable (int, char, …), ne može biti jedna od rezerviranih riječi, može počinjati sa slovom ili podvlakom.

**- što je paralelizam? podjela?**

Paralelizam je izvođenje više poslova istovremeno podjelom posla na manje cjeline ili podjelom posla po funkciji. Paralelizam se dijeli na paralelizam s više procesa, paralelizam s više dretvi i pseudoparalelizam.

**- što je Task?**

Task je najmanja cjelina posla Task Parallel biblioteke (Task Parallel Library - TPL).

**- razlike između parelelizma s više procesa i više dretvi**

Kod paralelizma s više procesa se stvara više instanci (procesa) istog programa koji međusobno komuniciraju slanjem signala te koriste dijeljenu memoriju, dok se kod paralelizma s više dretvi stvara više dretvi unutar programa gdje svaka dretva izvodi zadanu (obično sve istu) funkciju.

**- kako se ostvari paralelizacija u F#**

Pomoću:

* async { expression }
* let!
* Async.Parallel
* Async.RunSynchronously

**- sto su izrazi za inicijalizaciju klasa**

Konstruktori - posebne metode koje se zovu pri instanciranju objekata

**- blokovi s eksplicitnim granicama**

Eksplicitne granice se sastoje od početnog i završnog znaka ({ …. } za jezike familije C i Javu) unutar kojih blok može sadržavati deklaracije koje su lokalne za to područje.

**- jel F# jezik strogog tipa i kako se to vidi**

Da. Jezici strogog tipa onemogućuju korisniku (programeru) da “zaobiđe” sustav dodjele tipova, za razliku od npr. C-a gdje se svaki tip pokazivača može pretvoriti u bilo koji drugi običnim *castanjem*.

**- anonimni delegati i lambda izrazi - kako napisati i razlike**

Lambda izrazi su odsječci koda bez imena koji nisu unutar metode te su samostalni, dok su anonimni delegati eksplicitno definirani delegati koji nemaju ime te se odmah pri definiciji navode linije koda koje sadrži. Lambda izrazi se od C#-a 3.0 temelje na anonimnim delegatima te se najčešće unose kao predikati i projekcije.

**- izrazi za inicijalizaciju**

Alternativa konstruktorima, služe za inicijalizaciju javnih članova (svojstava ili varijabli) objekata te se inicijalizacija vrši u okviru jednog izraza.

**- Objasni apstraktne tipove u C#.**

Uvodi se tip kao parametar te se automatski određuje (T se zamjenjuje temeljem tipova ulaznih parametara). Kod se jednom piše te se just-in-time prevodilac brine da radi za različite (potrebne) tipove.

**- Načelo segregacije sučelja**

Korisnik (programer) ne smije biti prisiljen ovisiti o metodama koje koristi. Načelo segregacije sučelja dijeli velika sučelja u manja i specifičnija kako bi korisnici trebali poznavati metode koje su im potrebne.

**- Razlika između anonimnih delegata i lambda funkcija**

Lambda izrazi su odsječci koda bez imena koji nisu unutar metode te su samostalni, dok su anonimni delegati eksplicitno definirani delegati koji nemaju ime te se odmah pri definiciji navode linije koda koje sadrži.

**- razlika između sučelja i anonimnih delegata**

Sučelje predstavljaju apstraktne metode i svojstva koja mogu postojati samo u apstraktnim klasama te ga implementiraju izvedene klase za koje je ono zajedničko, dok za anonimne delegate nije potrebno pisati klasu s funkcijom koja će se zapakirati u delegat pa se može koristiti gdje je potrebno.

**- tip kod F#**

Konkretne tipove konstrukata u F# nije potrebno specificirati osim u rijetkim slučajevima kada prevoditelj ne može sam sa sigurnošću donijeti zaključak.

**- tri vrste povezivanja vrijednosti**

Plitko , duboko i statički

**- jesu li F# i Haskell strogog tipa, tj. koji od njih je?**

( oba )

**- što je imperativna paradigma?**

Imperativna paradigma temelji se na modelu Turingovog stroja, logicka na formalnoj logici, a funkcijska na lambda-racunu

**- currying**

od funkcije koja prima dva parametra napravimo funkciju koja prima jedan parametar tako da joj jedan damo unaprijed. npr. funkcija zbroji a b koja vraca zbroj brojeva a i b se moze iskoristiti da bi se napravila funkcija dodaj3 = zbroji 3. Funkcija dodaj3 je sad zapravo funkcija zbroji koja svaki put kao prvi parametar koristi broj 3.

**- Schonfinkelizacija**

Isto kao Currying, drugo ime.

**- Sto su funkcije prosirenja?**

To su funkcije ( metode ) koje proširuju neki tip javnom metodom. Moraju biti definirane kao static i public te moraju imati ključnu riječ this ispred oznake prvog parametra, taj tip parametra se ujedno i proširuje. Metode proširenja pružaju mogućnost nadogradnje javnog sučelja klase bez njegovih direktnih izmjena ili nasljeđivanja.

**- po čemu se razlikuje deklarativna od imperativne?**

• Imperativna paradigma

– Osnovna krakteristika je postojanje naredbe

– Obuhvata procedurlanu i objektno-orijentiranu paradigmu

• Deklarativna paradigma

– Osnovna krakteristika je nepostojanje naredbe

– Obuhvata funkcionalnu i logičku paradigmu

Imperativni program izvodi se (slijednim) izvodenjem naredbi, dok se deklarativni program izvodi vrednovanjem izraza

**- kako dobro napisati kod? (i što je meni najteže kad programiram? )**

Bla bla natucaj - reprezentativna imena ( varijabli, metoda, funkcija), intendacija, zagrade, inicijalizacija, (metode do 200 linija ), logičke cjeline, procedure, paziti na doseg varijabli,   
vrijeme zivota varijable da ne bude predugačko

**- kako imenovati varijable**

Varijable treba imenovati tako da njihov naziv bude reprezentativan, odnosno da se iz naziva može vidjeti čemu one služe, kako bi kod postao čitljiviji

**- sto je dependency inversion principle**

*A. High-level modules should not depend on low-level modules. Both should depend on* [*abstractions*](https://en.wikipedia.org/wiki/Abstraction_(computer_science))*.*

*B. Abstractions should not depend on details. Details should depend on abstractions*

**- lista u F#**

**let array1 = [| 1; 2; 3 |]**

**- apstrakcija**

• Modeliranje objekata na način da se koriste samo bitne komponente stvarnog objekta

**- enkapsulacija**

učahurivanjem (enkapsulacijom) klasa skriva neke atribute i neka ponašanja od ostalih klasa. slabom povezanošću objekti postaju neovisniji i interne promjene jednog objekta ne utječu na rad drugog

**- automatsko određivanje tipa**

Kod se jednom piše, a “just-in-time prevodilac” se brine da radi za različite (potrebne) tipove. Automatsko određivanje tipa (T je zamijenjen s int temeljem tipova ulaznih parametara)

**- Što su to sučelja?**

Sučelje je zapravo popis metoda neke klase, točnije popis javnih metoda kojima određena klasa može komunicirati s drugim klasama

**- Da li klasa može implementirati više sučelja?**

Klasa može implementirati više sučelja.

**- Da li klasa može naslijediti više klasa? Kako se taj problem rješava?**

-Kako u kojem jeziku (Java to ne dopušta, samo se jednu klasu može extendati ali može se implementirati više sučelja)

**- Odabrat neki princip (od ona 4) i objasniti.**

U programu je poželjno sve povezane programske elemente držati zajedno – PRINCIP BLISKOSTI

- open/closed principle

dependency inversion principle

**- Open/closed princip**

Razredi moraju biti zatvoreni za promjene, a otvoreni za proširenje - odnosno, funkcionalnost se dodaje npr. dekoratorima, ili metodama proširenja.

**- Polimorfizam.**

slična ponašanja različitih klasa • svaka klasa za sebe precizno definira zajedničko ponašanje

**- Koje su negativne posljedice polimorfizma?**

Komplicirano • Puno truda uloženo u dizajn • Sporije (ali samo kod aplikacija koje jako jako puno koriste takve objekte)

**- Sto je parametarski polimorfizam?**

Kako bi funkcije bile univerzalne razvojni inženjeri pišu “općenite” funkcije koje rade s bilo kojim tipom ulaznih varijabli. Uvođenje tipa kao parametra

**- Church - Rosserov teorem?**

ako se lambda izraz može svesti na dva različita lambda izraza M i N, onda postoji treći izraz Z do kojeg se može doći i iz M i iz N.

**- Kako se zove konačni izraz dobiven beta redukcijom?**

Višestrukom beta redukcijom izračunavamo vrijednost izraza i zaustavljamo se tek onda kada dalja beta redukcija nije moguća. • Tako dobijen lambda izraz naziva se normalni oblik i on intuitivno odgovara vrijednosti polaznog izraza.

**- Što su to delegati?**

Klasa (ili objekt) koja u sebi sadrži odsječak koda (metodu). Kao parametar delegat uzima ime metode – povratni tip i argumenti metode moraju odgovarati povratnom tipu i argumentima delegata

**- Što su to anonimni delegati?**

Delegat nema ime (anoniman je). Odmah pri definiciji se navode linije koda koje sadrži. Kompaktnija sintaksa pisanja delegata

**- Kako jednim pojmom obuhvatiti proceduralnu paradigmu i OOP?**

Imperativna paradigma**.**

**- Za funkcijske jezike kažemo da spadaju u :**

deklarativnu paradigmu.

**- Kako ispravno nazivati metode?**

Probati im nadjenuti ime po tome koji posao obavljaju

**Kohezija**

Koliko usko su operacije unutar postupka povezane. n Cilj:Jedan postupak za jedan i samo jedan zadatak.

**- Funkcionalna kohezija**

Najjača i najpoželjnija n Javlja se kada postupak obavlja jedan i samo jedan zadatak n Primjeri l sin(), GetCustomerName(), EraseFile() itd. n Pretpostavka: postupci rade točno ono što imena sugeriraju

**void funkcija je zapravo procedura**

DA

**Što je postupak (eng. routine)?**

Metoda ili procedura koja se poziva kako bi se obavio neki zadatak

**Sekvencijalna kohezija**

Postoji kada postupak sadrži zadatke koji se s razlogom moraju izvršiti u točno određenom redoslijedu

**Komunikacijska kohezija**

Postoji kada zadatci unutar postupka nisu ni na koji drugi način povezani osim što koriste iste podatke

**Vremenska kohezija** (

Postoji kada se zadaci ujedinjuju u postupak samo zato jer se izvršavaju u isto vrijeme )

**- singleton**

Jedinstveni objekt kako bi se stvorila samo jedna instanca tog objekta i dalje koristila u programu, primjer: samo jedna konekcija na bazu (da se ne spajamo svaki puta na bazu)

**- composite i decorator**

Kompozit omogućuje da grupom objekata manipuliramo kao s jednim objektnom, dekorator proširuje mogućnosti nekog postupka (metode)

**- što je apstraktna klasa**

Klase koje nije moguće instancirati (stvoriti objekte temeljem njih), Korisne za definiranje zajedničke funkcionalnosti

**- šta su anonimni tipovi kako nastaju i gdje se definiraju**

Klase koje se ne definira eksplicitno • Nastaju automatski u trenu izvođenja programa • Definiraju se automatski temeljem podataka koje će objekti tih klasa sadržavati

Anonimne metode (odsječci koda bez imena; koji nisu unutar metode; samostalni)

**- Liskov princip**

Izvedeni razredi moraju imati manju odgovornost od roditelja (Tipa ako roditelj vraća broj u rasponu -100 do 100 , onda dijete ne bi smjelo vraćat -200 do 200)

**- anonimni tip (karakteristike)**

Ne može ih se eksplicitno koristiti (npr. instancirati) u kodu (novostvorene klase su “nevidljive” za programera) • Objekti anonimnih tipova se mogu koristiti kao i bilo koji drugi objekti • Dva anonimna tipa su jednaka ukoliko im je i poredak, tip i naziv članskih varijabli isti • Prevodilac ne stvara više jednakih anonimnih tipova već iskorištava postojeće

**Boxing**

Pakiranje vrijednosnih tipova u referentne tipove

int i = 1; object o = i;

**- Koje kohezije su nepozeljne?**

Proceduralna kohezija l Logička kohezija l Slučajna kohezija

**- Sto su apstraktne metode? Gdje ih moramo instancirati,a gdje ne moramo? - - -**

• Predstavljaju sučelje – Implementiraju ga izvedene klase – Zajedničko za više izvedenih klasa. Ako je izvedena klasa apstraktna ne treba nadjačati apstraktne metode/svojstva osnovne klase – Ako izvedena klasa nije apstraktna treba nadjačati apstraktne metode/svojstva osnovne klase

**Sto su virtualne metode? kad njih moramo nadjacati?**

Virtualna metoda ne mora biti u apstraktnoj klasi – Virtualna metoda mora imati definiciju (linije koda) – Virtualna metoda se kod nasljeđivanja ne mora nadjačati

Slična je apstraktnoj metodi

**- Smiju li se apstraktne metode koristiti u neapstraktnim klasama?**

Klase koje nisu apstraktne ne mogu sadržavati apstraktne metode

**- princip bliskosti**

U programu je poželjno sve povezane programske elemente držati zajedno, Koristiti konstante gdje god je to moguće

**- ako imamo 2 jezgre, koliko bi bilo optimalno dretvi**

(kolega rekao dvije); mene je onda pitao ak imamo i treću koja stane, kako riješit problem čekanja (thread pool)

**- anonimni tipovi (tj. pitao je što važno kod polja anonimnog tipa –**

svi elementi moraju biti istog anonimnog tipa - naziv, tip, poredak članskih varijabli)

**- LINQ (što je + kako se stvara anonimni tip (select new + gore var))**

Omogućava pisanje upita nad listama objekata smještenima u memoriju. Upit se gradi temeljem blokova • Blokove određuju (filtriraju) metode proširenja. where, orderby i select

**- Razlika kod evaluacije f# i haskella**

(trenutna f#, haskell lijena)

**- metode prosirenja, sto je vazno kod njih (static public, this....)**

Proširenje nekog (bilo kojeg) tipa javnim metodama. Metode proširenja pružaju mogućnost nadogradnje javnog sučelja klase bez njegovih direktnih izmjena ili nasljeđivanja. Moraju biti definirane kao static i public. • Moraju biti unutar statičke klase. • Moraju imati ključnu riječ this ispred oznake tipa prvog parametra. Taj tip se ujedno i proširuje metodom. • Unutar njih je moguće koristiti samo metode javnog sučelja tipa koji se proširuje.

**- i medju prvim pitanjima je bilo sto je jedinstveno kod jezika kojeg sam opisivao u seminaru,**

pa ak mislite da ima nes jedinstveno navedite to, inace sam pucajte slicnosti sa drugim jezicima, prednosti i mane, samo pricajte... :)

**- što su apstraktne klase i moraju li im sve metode biti apstraktne**

Ne mogu se direktno koristiti (moraju se naslijediti) – Sadrže podatke – Sadrže metode

Javne metode i svojstva apstraktne klase se nazivaju i javnim sučeljem

**- override (nadjačavanje)**

Podklasa ne nasljeđuje varijablu članicu ili metodu svoje nadklase ako imaju isto ime. ako se radi o metodi, da *prerađuje* (eng. overriding) metodu svoje nadklase

**- Što je to aspektno orijentirano programiranje**

Aspektno orijentirano programiranje je programska paradigma koja povećava modularnost dodatnim razdvajanjem poslova (eng. concern) untar programa.

Aspektno programiranje dopunjuje OO programiranje uvođenjem dinamičkih elemenata u statičke modele.

**- Koji se problem javlja kod predubokog nasljeđivanja?**

Ako se koristi previše nasljeđivanja kod postaje težak za čitanje, razumijevanje, pračenje, održavanje, debuggiranje, itd. Tajođer vrijeme izvođenja se usporuje kako prilikom poziva se pozivaju sve metode nadklasa.

**- Duboko povezivanje (deep binding)**

Korištenje okoline koja je aktivna kada se stvara veza između stvarnog i formalnog parametra.

Koriste ga jezici sa statičkim i dinamičkim dosegom.

**- Closure- izvedba poziva po imenu- mehanizmi zatvaranja**

Prilikom poziva po imenu u funkciji se prenosi tekstualna reprezentacija izraza i okolina u kojoj se on izvršava. U trenu kada pozvana procedura treba izvršiti izraz, dohvaća se mehanizam za zatvaranje(closure) i izvršava se u okolini iz koje je funkcija pozvana.

**- Garbage collectors**

U jezicima bez eksplicitne dealokacije apstraktni je stroj nužno opremiti mehanizmima koji mogu automatski osloboditi memoriju koja se više ne koristi. Ima dvije faze: detekcija i kolekcija.

**- Implicitna polja**

Var ints = new[] {1,2,3,4}; Polje cijelih brojeva (implicitno određen tip) za vrijeme izvođenja programa.

**- Rekurzija repa (eng. tail recursion)**

-optimizacijska tehnika

-umjesto pamćenja međurezultrata, isti se šalje kao parametar funkciji

- moguća jedino kada se rekurzija poziva samo jednom unutar svake grane.

**- nepromjenjivost kod F#; možemo li nakon neke funkcije koja koristi x, rezultat spremiti u x**

Samo ako se koristi ključna riječ **mutable**, inače je varijabla nepromjenjiva.

**- Kada su dva apstraktna tipa jednaka?**

Kada su im jednaki tipovi, nazivi i poredak parametara