1. Korisćenjem primera, analizirati i napisati izveštaj o funkcionalnom programskom jeziku Haskel.
2. Ukratko objasniti sledeće modele programiranja:

* OOP programiranje
* Imperativno programiranje
* Proceduralno programiranje
* Funkcionalno programiranje
* Deklerativno programiranje

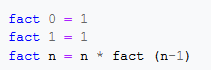
- Haskell

Haskel je funkcionalni, strogo tipizirani programski jezik opšte namene. Haskel koristi lenju evaluaciju, što znači da se izbegavaju nepotpuna računanja i potencira se modularizacija. Programi su koncizni i od 2 do 10 puta kraći od istih programa u drugim programskim jezicima. Haskel poseduje moćan sistem tipova kojima se veliki broj grešaka može otkriti pre samog izvršavanja programa. Podržava polimorfizam i preopterećenje. Podržava kompaktan i ekspresivan način definisanja listi kao osnovu strukture podataka.

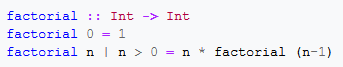
Haskel takođe naglašava upotrebu rekurzije koju dopunjuje mehanizmom podudaranja uzoraka, tačnije čuvarima (eng. guards). Funkcije višeg reda omogućavaju visok nivo apstrakcije i korišećenje funkcionalnih obrazaca. Poseduje razrađenu biblioteku standardnih funkcija i dodatnih modula.

U Haskelu ne postoji rezervisana, ključna, reč kojom se započinje deifnisanje funkcije.

- Primer funkcije koja izračunava faktorijel broja pomoću uparivanja šablona:



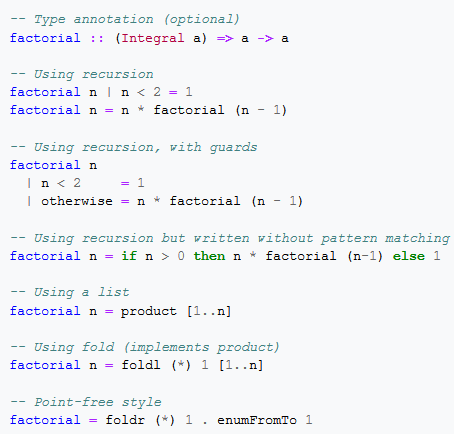
- Primer funkcije koja izračunava faktorijel broja:



Prva linija predstavlja deklaraciju funkcije i ona specificira da je factorial funkcija koja očekuje argument tipa Int i vraća rezultat tipa Int. U Haskelu je prisutno statičko tipiziranje, tačnije, kompajler je u stanju da automatski zaključi deklaracije, za skoro sve funkcije.

Ovo se događa čak i kada je deklaracija funkcije navedena, u tom slučaju kompajler proverava da li je deklaracija tačna.

Postoji više načina na koje možemo napisati funkciju u Haskel programskom jeziku, konkretno na primeru funkcije faktorijel:



U ovom primeru koristi se pisanje funkcije pomoću anotacija, rekurzije, rekurzije sa čuvarima, rekurzije bez uparivanja šablona, korišćenjem liste, implementacionih produkta i tačkom slobodnog stila.

Objektno-orijentisano programiranje

Objektno-orijentisano programiranje je paradigma programiranja, koja koristi objekte kao osovu za projektovanje računarskih programa i različitih aplikacija softvera. Predstavlja koncept programiranja koji koristi objekte, tj. instance klasa. Ove klase se sastoje od polja podataka i metoda zajedno sa njihovom interakcijom i služe za kreiranje aplikacija i računarskih programa.

Tehnike programiranja uključuju karakteristike kao što su:

* Apstrakcija podataka
* Enkapsulacija
* Razmena poruke
* Modularnost
* Polimorfizam
* Nasleđivanje

- Imperativno programiranje

Imperativno programiranje je koncept programiranja koji opisuje proces računanja uz pomoć sekvence izjava kojima se menja stanje programa. Kod imperativnog programiranja, program čini niz instrukcija računara. Osnovni alat koji omogućava ovaj način programiranje je uslovni skok. Imperativno programiranje često koristi dijagrame toka.

Imperativno programiranje opisuje računarski proces kao stanja programa i naredbi koje menjaju stanje. Stanje računarskog sistema je definisano sadržajem memorije i naredbama mašinskog jezika. Dakle, programi su skup naredbi koje računar treba da izvrši.

Hardverske implementacije računara su uglavnom imperativne, to znači da je hardver napravljen da izvršava mašinski jezik napisan u imperativnom stilu.

- Proceduralno programiranje

Procesuralno programiranje je lista ili skup instrukcija koje definišu računaru šta treba uraditi metodom korak po korak i kako se to obavlja od prvog koda na drugom kodu. To je programska paradigma izvedena iz struktuiranog programiranja izdvajanjem često ponavljanih blokova naredbi u procedure i njihovom izvršavanju na principu poziva procedure.

Procedure se sastoje od sekvence računarskih koraka koji se mogu izvršavati pozivom u bilo kojoj tački izvršavanja celog programa, uključujući slučaj izvršavanja i u dugim procedurama. Sam program time postaje niz poziva procedura, i koda koji povezje te pozive.

- Funkcionalno programiranje

Ovaj koncept programiranja tretira računanje kao evaluaciju matematičkih funkcija i izbegava definisanje stanja i promenljive podatke. Fokusira se na primenu funkcija koje, za razliku od imperativnog programskog stila, naglašavaju promene stanja. Bazira se na korišćenju matematičkih funkcija i izbegava stanja i promenljive podatke. Dakle, funkcionalno programiranje predstavlja programsku paradigmu kada se koriste funkcije, a ne objekti ili procedure, koje čine osnovne blokove programa.

Funkcionalno programiranje ima svoje korene u lambda kalkulusu, formalnom sistemu razvijenom za istraživanje načina definisanja funkcija, aplikacije funkcija i rekurzije.

Neki funkcionalni jezici u:

* Idris
* ML
* Haskell

Funkcionalno programiranje karakterišu objekti koji su konstante ili funkcije, red izvršavanja operacija nije sofisticiran i nedefinišu se stanja.

- Deklerativno programiranje

Programiranje zasnovano na pravilima pri čemu je svako pravilo nezavisno i redosled po kojim su pravila definisana ili izvršena nisu značajna. Za razliku od imperativnog programiranja, deklerativno programiranje izražava ono što program treba da ostvari bez propisivanja kako to učini u smislu sekvence akcija koje treba obaviti.

Deklerativno prograiranje izvršava računanje nezavisno od drugih stanja računanja. Nema stanja, svaka akcija je nezavisna od prethodne. Deklerativno prograiranje je determinističko, isti ulaz uvek vraća isti rezultat.

Deklerativno programiranje može biti funkcionalno i logičko programiranje.