PJISP – bitne teze za ispit (moguca pitanja)

- Programski jezik C radi na principu *imperativne paradigme*.
- Bitna odlika C programskog jezika jeste *prenosivost izvornog koda*.
- *Algoritam* jeste opis obrade podataka, sa preciziranim operacijama koje se izvrsavaju prema odredjenom redosledu.
- Obavljanje obrade podataka se zapravo svodi na *izvrsavanje njenog algoritma*.
- Algoritam izrazen u programskom jeziku naziva se *program*, a izrada programa je *programiranje*.
- Svaki program mora imati bar jednu funkciju, *main( ) funkciju*, od koje pocinje izvrsavanje programa. Sve funkcije su ravnopravne i mogu pozvati jedna drugu. Sastoje se od naredbi.
- Naredbe se grupisu u takozvane *blokove*, koji se obelezavaju sa { }.
- *Program u C-u*: izvorna datoteka + datoteka zaglavlja.
  Izvorna datoteka: 1) pretprocesorske direktive; 2) globalne deklaracije; 3) definicije funkcija.
- *Identifikatori* su nazivi/imena struktura ili promenljivih. Prvi znak identifikatora ne sme biti cifra! Maksimalan broj karaktera u identifikatoru je po standardu 31. Ali to se cesto ne postuje.
- *__func__* je funkcija koja pristupa nazivu zeljene funkcije.
- Oblasti vazenja identifikatora: 1) file scope (oblast vazenja fajl) 2) block scope (blok) 3) function prototype scope (prototip) 4) function scope (funkcija).
- *_Bool* je neoznaceni celobrojni tip!
- Pomoc za opseg: operator *sizeof ()* i datoteka *limits.h*
- *Float* – 4 bajta, preciznost 6 cifara; *Double* – 8 bajtova, 15 cifara; *Long double* – 10 bajtova, 19 cifara.
- Konstante i promenljive nikada ne mogu biti tipa void!
- *Operandi* i *operatori*. Operatori * i / imaju veci prioritet u odnosu na operatore + i - .
- Logicki netacan izraz ima vrednost 0, dok je vrednost logicki tacnog izraza uvek razlicita od 0 (pre svega 1).
- Uglaste zagrade *[ ]* su binarni operator. Operatori *--* i *++* su unarni operatori. Binarni operatori *==*, *!=*, *<*, *>*, *<=*, *>=*. Binarni operatori su *&&* i *||*, a unarni *!* .
- Ako operatori nisu istog tipa, tada se vrsi implicitna konverzija uzeg u siri tip.
- Unarni operatori imaju visi prioritet od binarnih. Aritmeticki operatori > relacioni operatori > logicki operatori.
- *Operatori sa bocnim efektima* su operatori dodele i uvecanja i umanjenja.
- *Operator dodele ''=''* : levi operand – naziv promenljive, desni operand – izraz.
- *Niz* je serija objekata istog tipa. Definicija niza: TIP IME [BROJ_ELEMENATA];
- *Najmanja memorijska lokacija* koja moze da se samostalno adresira je obicno jedan bajt, koji sadrzi 8 bitova.
- *Pokazivac* – prost podatak, u njega se smesta adresa neke lokacije u memoriji, obicno zauzima 4 bajta.
- *Referenciranje* – p = &a, sada pokazivac p pokazuje na promenljivu a.
  Prefiksni unarni operator & ne sme da se primeni na prolazne podatke, mora na stalne memorijske podatke.
- *Dereferenciranje* – pristupanje nekom podatku preko njegove memorijske adrese, uz pomoc unarnog operatora *.
- *NULL* – specijalna konstanta, ako *p = NULL, onda p ne pokazuje ni na sta.
  Neinicijalizovani pokazivaci ne pokazuju na NULL! Moraju se eksplicitno inicijalizovati na NULL!
- *Pokazivac tipa void **- moze da pokazuje na adresu bilo kog podatka, ali ne i na njegov tip (objekti tipa void ne postoje). Da bi pristupili tipu tog objekta, vrsimo cast-ovanje pokazivaca na taj tip.
- *Promenljivi pokazivac na nepromenljive podatke* – npr. const int *p. (nepromenljivi podaci tipa int)

- *Nepromenljivi pokazivac na promenljive podatke* – npr. int *const p = &k.
- *Nepromenljivi pokazivac na nepromenljive podatke* – npr. const int *const p = &k.
- *Dodeljivanje adrese jednog pokazivaca drugom* – operatorom = moze da se dodeli vrednost jednog pokazivaca drugom, ako su njihovi tipovi isti. Ako im nisu isti tipovi, potrebno je usput izvrsiti cast-ovanje. Cast-ovanje nije neophodno ako je jedan od pokazivaca tzv. genericki pokazivac tj. tipa void *.
- *Sabiranje i oduzimanje celobrojnog podatka* – operatorima +, ++ i += (odnosno -, -- i -=) moze da se izracuna zbir (razlika) vrednosti pokazivaca i celobrojnog podatka. Pokazivac ne sme biti tipa void *.
- Jedinica mere pri sabiranju (oduzimanju) vrednosti pokazivaca i celobrojnih podataka je *velicina pokazivanih podataka*, a ne tip celobrojnog elementa. (npr. p pokazuje na neki element niza, p+1 pokazuje na naredni, itd.)
- *Oduzimanje i uporedjivanje pokazivaca* – Pre svega, pokazivaci moraju biti istog tipa!
  Dozvoljeno je oduzimanje dva pokazivaca istog tipa (razlicitog od void *) operatorom - .
  Dozvoljeno je uporedjivanje relacijskim operatorima <, >, <=, >= dva pokazivaca koji pokazuju na elemente istog niza.
  Dozvoljeno je uporedjivanje dva pokazivaca operatorima == i != (mogu biti i tipa void *). Ovim se ispituje da li pokazivaci pokazuju na isti podatak ili ne. Dozvoljeno je uporedjivanje pokazivaca (moze biti i void *) sa NULL-om. Tako se ispituje da li pokazivac uopste pokazuje na neki podatak.
- *Pokazivac i niz* – int a[20], int *pa. Izraz pa = a; je isto sto i pa = &a[0]; p pokazuje na prvi element (memorijsku lokaciju) niza a.
- *Identifikator niza* je nepromenljivi pokazivac na podatke niza, ciji je tip jednak tipu elemenata niza.
- *Ekvivalentni izrazi –* 1) &a[i] ←→ a+i ←→ i+a;
  2) a[i] ←→ *(a+i) ←→ *(i+a) ←→ i[a];
- *Ograniceni pokazivac –* pokazivac sa kvalifikatorom restrict. Veza izmedju ovakvog pokazivaca i podatka na koji on pokazuje je sledeca: tokom zivotnog veka pokazivaca ovom podatku se moze pristupiti (ili on se moze menjati) samo preko njegovog pokazivaca.
- *Parametri funkcije –* to su lokalne promenljive, oblast vazenja im je uvek samo funkcija.
- *Prototip funkcije -* isto sto i deklaracija funkcije, opisuje interfejs funkcije.
- *Umetnute funkcije (inline) –* cesto upotrebljivanje ovih malih funkcija (posebno ako sadrze cikluse) usporava rad programa i kompajliranje. Obicno se trazi od programa da na tim mestima umetne masinski kod umesto inline funkcija.
- *Rekurzivne funkcije -* funkcije koje pozivaju same sebe (direktno/indirektno). Mora postojati uslov za zavrsetak pozivanja (rekurzije).
- *Operator clana strukture –* to je zapravo tacka ''.'' (npr. struct.ime). Ovaj operator je u samom vrhu prioriteta, zajedno sa ( ) (funkcija) i [ ] (niz).
- *Pristup elementu strukture preko pokazivaca –* (*p).m ili laksi (i cesci) nacin p->m.
- *Datoteke –* predstavljaju sekvence bajtova, funkcijom fopen ( ) se inicijalizuje objekat tipa FILE i povezuje se s tokom podataka. Prenosenje podataka u odnosu na buffer moze se odvijati na 3 sledeca nacina:
  1) po punjenju bafera (fully buffered) – znaci u baferu se normalno prenose samo ako je on pun.
  2) posle znaka za novi red (line buffered) – znaci se normalno prenose samo ako je u bafer upisan znak za novi red ili ako je on pun.
  3) izvan bafera (unbuffered) – znaci se prenose sto je brze moguce.
- *Standardni tokovi podataka –* stdin, stdout, stderr. Stdin je obicno povezan s tastaturom, a ostala dva sa ekranom. Te veze se mogu izmeniti tzv. preusmeravanjem uz pomoc funkcije freopen ( ).

- *Otvaranje datoteke –* najcesce preko fopen ( ) (za parametre ima string objekte/tokove podataka), ali nekada i preko freopen( ) ili tmpfile( ).
- *Rezim pristupa –* ''r'' – read (za citanje), ''w'' – write (za pisanje), ''a'' – append (za dopisivanje).
- *Unija –* je struktura koja moze cuvati (u razlicito vreme) objekte razlicitih tipova i velicina. Unija zauzima samo onoliko memorije koliko je potrebno njenom najvecem clanu, pa se stoga moze u svakom trenutku pristupiti samo jednom clanu unije.
- *Polja bitova –* pomocu njih se u C-u postize kompaktnost memorije. Polja bitova se deklarisu samo kao oznaceni ili neoznaceni celobrojni tipovi. Pojedinacna polja bitova nemaju adrese!
- *Heap -* podrucje nezauzete memorije, koja se na zahtev dodeljuje procesu.
- *Funkcije za dinamicku memoriju -* : sve se nalaze u <stdlib.h> biblioteci.
  1) malloc – dodeljuje nov blok u memoriji. Vraca adresu prvog bajta u bloku na koji smo primenili sizeof( ). primer malloc f-je: malloc(sizeof(CVOR));
  2) calloc – dodeljuje nov blok u memoriji. Void *calloc(n , size) rezervise memorijski blok (inicijalizovan na 0) dovoljan za memorisanje n podataka velicine size, tj. n*size.
  3) realloc – menja velicinu dodeljenog bloka. Oslobadja memorijski blok i rezervise novi velicine size bajtova. Void *realloc(p, size), gde je p pokazivac na blok koji se realocira, a size velicina tog izmenjenog bloka.
  4) free – oslobadja dodeljenu memoriju. Free(arg), gde arg mora biti pokazivac na pocetak bloka koji oslobadjamo tj. dealociramo.

  *u slucaju neuspesnog alociranja memorije, prve 3 funkcije vracaju NULL.
  Sve 3 su oblika void *malloc/calloc/realloc.
- Prilikom definisanja skalarnih promenljivih ili nizova, memorija se zauzima staticki!
- *Dinamicka alokacija memorije* izvrsava se u vreme izvrsavanja programa.
- *Genericki pokazivac (tipa void *)* sadrzi samo adresu, ne i tip podatka na koji pokazuje.
- *Strukture podataka* – linearne: dinamicki niz, spregnuta lista, stek, red, dek.
  -- nelinearne: stablo, graf.
- *Dve vrste fizicke realizacije linearne strukture podataka* – sekvencijalna fizicka i spregnuta fizicka.
- *Niz*: jednodimenzionalni (vektor) i visedimenzionalni (matrica). Za smestaj niza u memoriji se koristi sekvencijalna reprezentacija.
- *Dinamicka matrica*: dinamicki niz pokazivaca na dinamicki niz elemenata. U prevodu, matrica A[i][j] se moze zapisati i ovako *(*(A+i) + j).
- Kod sekvencijalne realizacije uzastopni elementi su nuzno i susedni u memoriji, dok kod spregnute realizacije mogu biti *sa bilo kog mesta u memoriji*.
- *CVOR* je samoupucujuca struktura spregnute liste.
- *Spregnuta lista*: LISTA { CVOR *glava; }; CVOR { bilo_koji_tip INF; struct cvor_st *sledeci; }; .
- *Jednostruko spregnuta lista*: uredjeni par P = (S(P), r(P)). Moguce je dodavanje i uklanjanje pojedinacnog elementa na bilo kojoj poziciji. Uredjena lista – ako se po sadrzaju INF-a moze napraviti funkcija za poredjenje i ako su elementi liste sortirani, u suprotnom neuredjena lista. Poslednji element ovakvih lista uvek pokazuje na NULL. Spregnuta lista moze da sadrzi promenljivi broj cvorova.
- *Kruzna jednostruko spregnuta lista*: poslednji cvor pokazuje na glavu (pocetak) liste. Ovakve liste poseduju osobinu simetricnosti. Operacija CONCATENATE(glava1, glava2) je operacija spajanja 2 kruzne liste i ne zahteva prolazak do kraja prve liste, jer glava1 pokazuje vec na kraj prve liste, a kasnije i na kraj spojene liste. **Zaglavlje liste** – poseban cvor koji pokazuje na prvi cvor u listi, razlikuje se po posebnoj vrednosti u polju info.

- *Dvostruko spregnuta lista*: ili simetricna lista (za par (a,b) se uvodi veza koja 'slika' (b,a)). Ovakve liste mogu biti u cirkularnom obliku. Moze se posmatrati kao unija dve pojedinacne jednostruko spregnute liste.
  Cvor ovakve liste sadrzi dva pokazivaca - *sledeci i *prethodni. Definicija DS liste jeste DP=(S(DP), r(DP)), gde se r moze rasclaniti na r1 i r2, ali mora da vazi r= r1 U r2 i r1-presek-r2 = ∅ i da strukture (S(DP), r1(DP)) i (S(DP), r2(DP)) budu jednostruko spregnute liste.
  CVOR { bilo_koji_tip KLJUC; struct cvor *sledeci; struct cvor *prethodni; }; .
  Prvi cvor u polju *prethodni i poslednji cvor u polju *sledeci pokazuju na NULL.
  Prednost DS liste: dodavanje cvora se moze izvrsiti ispred ili iza zeljenog selektovanog cvora.
  LISTA { CVOR *prvi; CVOR *poslednji; }; .
- *Multi-lista*: organizuje podatke po vise relacija u vise dimenzija, broj glava (pokazivaca na prvi element liste po nekoj relaciji) mora odgovarati broju relacija. Visestruka unija jednostrukih lista.
- *Visestruka lista*: NIJE visestruko spregnuta lista! Kod ove liste ne moraju svi elementi da ucestvuju u svim relacijama. VP=(S(VP), r1,...,rn), gde je n>2. Svaki element mora imati prostor (memorijski) za sve pokazivace svake pojedinacne relacije, cak i ako nije u toj relaciji.
- *Sortiranje liste*: MERGE-SORT! 3 koraka: 1) lista se deli na 2 manje liste.
  2) rekurzivno sortiranje te dve liste.
  3) konkatenacija te dve liste u sortiranu originalnu listu.

- *STEK*: LIFO tip (last in = first out). Dozvoljen je pristup samo prvom elementu (*ptop) i moze se dodati novi element samo ispred prvog elementa (tj. vrha steka).
  Realizacija steka moze biti sekvencijalna ili spregnuta (po potrebi). Stek ima jedan pokazivac (*ptop).
- *SEKVENCA*: definicija jeste D=(S(D), r(D)). Struktura je linearna. Elementi se mogu ukloniti samo svi odjednom. Novi element se dodaje iskljucivo na kraj sekvence. Pristup je moguce svakom elementu sekvence, ali pocevsi od onog kome se poslednji put pristupalo.
- *RED*: FIFO tip (first in=first out). Pocetak (terminal na kom se brise element) i kraj (terminal na kom se dodaje novi element u RED) RED-a. Indeks pocetka (gde treba da se obavi sledeca operacija UPISI) i indeks kraja (gde treba da se obavi sledeca operacija PROCITAJ). Ako su indeksi kraja i pocetka jednaki – RED je prazan, nema sta da se procita. Fizicka struktura STEK-a je zapravo specijalan slucaj fizicke strukture RED-a, gde je *poslednji = Amin = const. Sekvencijalna struktura RED-a ima posebnu osobinu: **stanje prepunjenosti RED-a (nastaje prvom primenom InQueue funkcije)**, ciji memorijski prostor nije skroz zauzet.
  Mehanizam za pomeranje reda se obezbedjuje cirkularnom strukturom.
- *Kruzni RED*: npr. bafer za tastaturu personalnog racunara. Indeks kraja se inicijalizuje na jedno mesto iza indeksa pocetka i nije mu dozvoljeno da ga stigne.
- *DEK*: pristup, dodavanje i uklanjanje su dozvoljeni na oba kraja. Prvi i poslednji ne postoje, vec krajnji levi i krajnji desni. Za spregnutu realizaciju je potrebno dvostruko sprezanje.
- *STABLO*: **Listovi** – cvorovi koji nemaju sledbenike (podredjene cvorove). **Grana** – uredjeni par cvorova. Kljucna rec koja opisuje stablo je **hijerarhija**. **Lanac** – skup cvorova povezanih granama. **Put** – orijentisani lanac. **Slabo povezan digraf** – svaka dva cvora povezana jednim lancem. **Visina stabla** – broj cvorova na najduzem putu.
- *Tipovi stabla*: 1) kompletno (reda n) – svi elementi sem listova imaju izlazni stepen n.
  2) puno – svi putevi od korena do listova su iste duzine.
  3) balansirano – stablo gde se broj elemenata podstabla na istom hijerarhijskom nivou razlikuje najvise za 1.
- *Binarno stablo pristupa*: levo dete sadrzi < vrednost od roditelja, a desno dete > vrednost od roditeljska.

- *Degenerisano stablo*: za svaki roditeljski cvor postoji samo jedno dete (jedan podcvor).
- *Binarni hip*: binarno stablo sa 2 dodatna ogranicenja – svojstvo oblika, svojstvo hipa.
  Up-heap: dodavanje elementa na donji nivo stabla, ako su roditelj i dete u ispravnom redosledu – stop, ako nisu – zamenimo im mesta i vratimo se na prethodni korak.
  Down-heap: zamena korena hipa za poslednji element na poslednjem nivou stabla, zatim se uporedi novi koren sa svojom decom i ako su u ispravnom redosledu – stop, ako nisu – zamena mesta sa jednim od dece.
  Prilikom implementacije binarnog hipa, pokazivaci nisu potrebni (koristi se aritmetika sa indeksom clana niza).
- *Heap-sort*: izdvajanje maksimalnog elementa i stavljanje na poslednje mesto, izdvajanje maksimalnog elementa od preostalih i stavljanje na pretposlednje mesto, itd. , sve dok se niz ne uredi.
- *Proces* – element dijagrama toka sa jednom ulaznom i jednom izlaznom linijom u kome se obavlja obrada ili prenos podataka.
- *Predikat* – element dijagrama toka sa jednim ulazom i dva izlaza, izvrsava iskljucivo kontrolnu funkciju. Izracunava vrednost ulaza i proces dalje upucuje prema true ili false izlazu.
- *Kolektor* – element dijagrama toka sa 2 ulaza i 1 izlazom, ne vrsi obradu, vec sluzi za usmeravanje 2 izvora obrade u 1 zajednicki tok.
- *Pravilan program* – mora da ispunjava 3 uslova: da ima 1 ulaznu i 1 izlaznu liniju, da za svaki cvor postoji putanja koja ide od ulazne linije kroz taj cvor do izlazne linije.
  Pravilan program ne sme sadrzati u sebi beskonacne petlje i izolovane (nedostizne) programske segmente.
- *Pravilan potprogram* – svaki sastavni deo pravilnog programa koji je i sam pravilan program.
  Svaki proces je potprogram. Predikati i kolektori nisu potprogrami.
- *Prost program* – pravilan program ciji svaki pravilni potprogram ima samo jedan cvor.
- *Baza strukturiranih podataka* – skup prostih programa cijom superpozicijom mozemo doci do realizacije proizvoljnog pravilnog programa. Baza je **dovoljna** (ako se izostavljanjem neke strukture dobije podbaza) ili **potrebna** (ako nije dovoljna).
- *Formalno strukturiran program* – svaki pravilan program sacinjen jedino od kontrolnih struktura jedne baze.
- *Ekvivalentni programi* – kada za iste ulazne podatke generisu iste rezultate (izlaze).
- *Funkcija veze* – F(i, j) odredjuje vezu od cvora i do cvora j, gde ako je F(i,j)=T (true) veza postoji, a F(i,j)=F (false) veza ne postoji. Za svaki i postoji samo jedan j takav da je F=T.
- *Kanonicna forma programa* – WHILE-CASE ili REPEAT-CASE forma programa.
- *3 osnovne kontrolne strukture* – sekvenca, selekcija, iteracija.
- *Osobine algoritma*: 1) diskretnost – ne postoje kontinualni algoritmi, svi su diskretni.
  2) rezultativnost – nakon konacnog broja koraka, generise se rezultat.
  3) determinisanost – za iste ulazne podatke daje isti rezultat.
  4) elementarnost – zakon dobijanja izlaznih velicina mora biti jasan i prost.
  5) masovnost – algoritam mora biti primenljiv na mnostvo ulaznih podataka.
- *Kompleksnost algoritma* – vreme rada (ili broj koraka potrebnih za dolazenje do kranjeg cilja tj. resenja).
- *Tjuringova masina* je prebrojiv skup. Radi nad konacnim skupom simbola.
- Skup svih algoritamski resivih problema je prebrojiv.
  Skup svih problema odlucivanja je neprebrojiv.
  Postoje problemi za koje ne postoje algoritmi.
- *Algoritamske seme*: linijske (dele se na proste i razgranate) i ciklicne.
- *Proste*: svaki korak algoritma se izvrsava tacno jednom u toku izvrsavanja samog algoritma.

- *Razgranate*: svaki koram se izvrsava tacno jednom i obavezno sadrzi bar 1 uslovni algoritamski korak.
- *Izlazni kriterijum ciklusa*: uslov za izlazak iz ciklusa (kod ciklicnih algoritamskih sema).
- Ne postoji program bez gresaka. Program koji nije testiran, nije ni zavrsen.
- *Korektnost programa* – mera u kojoj program zadovoljava specifikaciju. (za proveru uskladjenosti programa)
- *Robusnost programa* – imunost na razne sadrzaje, pre svega na neregularne (neocekivane) podatke. (za proveru ponasanja u eksploatacionim uslovima)
- *Defekt (fault)* je rezultat greske koji se pojavljuje u programskom kodu. Kada se u toku izvrsavanja programa naidje na fault, dolazi do **otkaza (failure)** programa. Eksplicitan prikaz failure-a se naziva **incident**.
- *Test case (testna stavka)* – primerak skupa ulaznih vrednosti preko kojih se program moze pokrenuti. Skup test case-ova se zove **test set (test skup)**.
- *Pristupi testiranju*: konstruktivni (dokaz da program nema gresaka) i destruktivni (dokaz da ima gresaka).
- *Tipovi destruktivnog pristupa*: 1) **analiza programa**: analiza programskih segmenata (Code Inspection) i simulacija izvrsavanja programa (Walkthroughs).
  2) **White Box Strategy** (ili strukturno testiranje).
  3) **Black Box Strategy** (ili funkcionalno testiranje).
- *Specifikacija programa* mora biti tacna, razumljiva i potpuna.
- *Black box strategy*: 1) metoda granicnih vrednosti, 2)metoda klasa ekvivalencije.
- *Single fault assumption*: uslov da otkaz ne nastaje uticajem vise defekata, vec samo jednog.
- **Sistemski test**: testiranje programa u celosti.
- *Inkrementalno testiranje*: 1) **top-down** („s vrha ka dnu“, krece od glavnog modula, ide od nadredjenih prema podredjenim modulima), 2) **bottom-up** („od dna ka vrhu“, krece od najnizeg modula, ide od podredjenih kad nadredjenima).