Programiranje 1

- odgovori na pitanja iz skripte -

Mina Milošević
mi17081@alas.matf.bg.ac.rs

2017/2018

1.1. Nabrojati osnovne periode u razvoju računara i navesti njihove osnovne karakteristike i predstavnike.

Premehanicki period (3000g.p.n.e. - 1450g.n.e.)

- Pojava prvog alfabeta i pisma (oko 3000 g. p.n.e. u Mesopotamiji)
- Formiranje ne-slikovnih alfabeta (Fenicani, Grci, Rimljani,...)
- Koriscenje papira za pisanje (Kina, oko 100 g. n.e.)
- Pojava prvih biblioteka
- Razvoj brojcanih sistema (nepozicionih i pozicionih)
- Pojava i koriscenje abakusa kao sredstva za racunanje

Mehanicki period (1450g. - 1840g.)

- 1450. - pojava stamparske prese sa pokretnim slogovima od metala (Gutenberg)
- 1614. - uvodjenje logaritama i decimalne tacke u zapisima brojeva (Dzon Neper)
- 1622. - pojava klizajuceg lenjira - sibera (Viljem Outred)
- 1623. - konstrukcija masine za sabiranje i oduzimanje (Sikard)
- 1642. - Paskalina – mogla je da sabira i oduzima osmocifrene brojeve
- 1673. - pojava masine koja je mogla da vrsi sabiranje, oduzimanje, mnozenje i deljenje (Lajbnic)
- 1801. - prva upotreba busene kartice I prva programabilna masina (Zakardov automatski razboj)
- Masine Carlsa Bebidza (diferencijska – koristila metod konacnih razlika, nacrt 1822, prototip 1832, nacrt analiticke masine 1833. godine; Ada Bajron je napisala prve programe za Bebidzovu masinu)

Elektromehanicki period (1840g. - 1939g.)

- Razvoj telekomunikacija (telegraf 1830., telefon 1876., radio 1984.)
- 1854. - pojava Bulove algebre
- 1884. - konstrukcija automatske masine za tabuliranje zasnovane na busenim karticama (Herman Holerit)
- Razvoj razlicitih elektromehanickih kalkulatora u prvoj polovini 20. veka
- Pojava elektromehanickih racunara specijalne nemene
- 1941. - Konrad Cuze konstruisao je 22-bitni uredjaj za racunanje Z3 koji je imao mogucnost koriscenja petlji
- Obrada podataka u udaljenom okruzenju
- Konstrukcija MARK I elektromehanickog racunara (zavrsen 1944. godine)

Elektronski period (1939g. - danas)

- 1936. – Tjuringova masina – uproscena verzija savremenog racunara
- 1939. - ABC kalkulator za resavanje sistema jednacina (Atanasov i Beri)
- 1943. - Kolos (Tjuring) – dekriptovanje nemackih sifrovanih poruka
- 1946. - ENIAC – racunar zasnovan na dekadnom sistemu (Ekert i Musli)
- 1945. - EDVAC (nacrt) – mogucnost cuvanja programa u memoriji zajedno sa podacima i kasnije izvrsavanje. Konstruisan je 1951.
- 1949. - EDSAC – konstrusian po fon Nojmanovoj arhitekturi
- 1949. - BINAC – prvi racunar sa dualnim procesorom
- 1950. - Vihor – racunar namenjen radu u realnom vremenu

1.2. Kojoj spravi koja se koristi u današnjem svetu najviše odgovaraju Paskalove i Lajbnicove sprave?

Digitron/kalkulator.

1.3. Ko je i u kom veku konstruisao prvu mehaničku spravu na kojoj je bilo moguće sabirati prirodne brojeve, a ko je i u kom veku konstruisao prvu mehaničku spravu na kojoj je bilo moguće sabirati i množiti prirodne brojeve?

Blez Paskal je prvi konstruisao 1642. godine mehanicku masinu koja je mogla da sabira i oduzima prirodne brojeve, masine su se zvale Paskaline. Lajbnic je prvi konstruisao masinu koja je mogla da radi sve 4 aritmeticke operacije 1672. godine.

1.4. Kakva je veza između tkačkih razboja i računara s početka XIX veka?

Bili su programibilni.

1.5. Koji je znacaj Carlsa Bebidza za razvoj racunarstva i programiranja? U kom veku je on dizajnirao svoje racunske masine? Kako se one zovu i koja od njih je trebalo da bude programabilna? Ko se smatra prvim programerom?

Dizajnirao je svoje masine u prvoj polovini XIX veka, njegova masina je bila sa namerom da moze da se programira putem busnih kartica. Omogucavao je sekvencijalno izvrsavanje naredbi, grananja i skokove. Masine su: diferencijska masina (nije zavrsena) i analiticka.
Ada Bajron je smatrana prvim programerom, zajedno sa Bebidzom je napisala prve programe za njegovu Analiticku masinu.

1.6. Na koji nacin je Herman Holerit doprineo izvrsavanju popisa stanovnika u SAD 1890? Kako su bili cuvani podaci sa tog popisa? Koja cuvena kompanija je nastala iz kompanije koju je Holerit osnovao?

Napravio je uredjaj za citanje busnih kartica, sa 10 godina potrebnih za obradu popisa je smanjio vreme na svega jednu godinu. Podaci su se zapisivali na busne kartice prilikom popisa na masinski citljivom medijumu (busnim karticama). Horleritova kompanija je kasnije postala IBM.

1.7. Kada su nastali prvi elektronski racunari? Nabrojati nekoliko najznacajnijih.

Nastali su krajem 1930-ih.
• ABC (resavanje linearnih jednacina)
• Kolos (desifrovanje nemackih poruka)
• ENIAC (racunanje trajektorije projektila)

1.8. Na koji nacin je programiran racunar ENIAC, a na koji racunar EDVAC?

ENIAC se reprogramirao time sto su intervenisali na preklopnicima i kablovima, proces je mogao da traje nedeljama. EDVAC je racunar sa skadistenjem programa u memoriji i programiran je ucitavanjem programa iz glavne memorije; programiran je softverski.

1.9. Koje su osnovne komponente racunara fon Nojmanove arhitekture? Sta se skladisti u memoriju racunara fon Nojmanove arhitekture? Gde se vrsi obrada podataka u okviru racunara fon Nojmanove arhitekture? Od kada su racunari zasnovani na fon Nojmanovoj arhitekturi?

Osnovne komponente su CPU i glavna memorija koji su medjusobno povezani. U memoriji se skladiste podaci koji se obradjuju, ali i programi, predstavljeni nizom elementarnih instrukcija. Napisani su kao obicni binarni zapis. Sva obrada podataka se vrsi u procesoru. Od kraja 1930-ih su zasnovani.

1.10. Sta su to racunari sa skladistenim programom? Sta je to hardver, a sta softver?

Racunari sa skladistenim programom su racunari u kojima je vidljiva podela na hardver i softver.
Hardver je opipljivi deo racunara (delovi koji cine racunar), a softver su programi i prateci podaci koji odredjuju izracunavanja koje vrsi racunar.

1.11. Sta su procesorske instrukcije? Navesti nekoliko primera.

Procesorke instrukcije su komande koje procesor razume: sabiranje, oduzimanje, mnozenje, deljenje, poredjenje.

1.12. Koji su uobicajeni delovi procesora? Da li se u okviru samog procesora nalazi odredena kolicina memorije za smestanje podataka? Kako se ona naziva?

Delovi procesora su Control Unit koja upravlja njegovim radom i aritmeticko-logicka jedinica koja vrsi operacije sabiranja, oduzimanja, mnozenja, deljenja, poredjenja. Postoji manji broj registara koji privremeno mogu da cuvaju podatke. Naziva se Cache memorija.

1.13. Ukratko opisati osnovne elektronske komponente svake generacije racunara savremenih elektronskih racunara? Sta su bile osnovne elektronske komponente prve generacije elektronskih racunara? Od koje generacije racunara se koriste mikroprocesori? Koji tipovi racunara se koriste u okviru III generacije?

I - Vakumske cevi
II - Tranzisotri
III - Integrisano kolo
IV - Mikroprocesor
Osnovne komponente prve generacije racunara su vakumske cevi, logicka kola i magnetni dobosi. Od 4 generacije. Mainframe racunari se najvise koriste.

1.14. U kojoj deceniji dolazi do pojave racunara za kucnu upotrebu? Koji je najprodavaniji model kompanije Commodore? Da li je IBM proizvodio racunare za kucnu upotrebu? Koji komercijalni kucni racunar prvi uvodi graficki korisnicki interfejs i misa?

U XX veku, krajem 1950. do polovine 1960ih. Najprodavaniji je Commodore 64. Jeste, racunar IBM 1401 je bio dominantan, sa vise od trecine trzista. Apple-ov Macintosh.

1.15. Koja serija Intel-ovih procesora je bila dominantna u PC racunarima 1980-ih i 1990-ih godina?

IBM serija x86 je bila dominantna.

1.16. Sta je to tehnoloska konvergencija? Sta su to tableti, a sta pametni telefoni?

Konvergcija podrazumeva spajanje vise razlicitih uredjaja u jedinstvene celine, tableti i pametni telefoni . Pametni telefoni su mobilni telefoni koji imaju vise naprednih opcija i konektivnosti od obicnih telefona.

1.17. U koju grupu jezika spadaju masinski jezici i asemblerski jezici?

Nizi programski jezici, jezici razumljivi racunaru.

1.18. Koje su osnovne komponente savremenog racunara? Sta je memorijska hijerarhija? Zasto se uvodi kes-memorija? Koje su danas najkoriscenije spoljne memorije?

Osnovne komponente: procesor, glavna memorija, periferijski uredjaji.
Memorijska hijerarhija sluzi da se unapredi funkcionisanje sistema, ide od najmanje (ali i najbrze CPU registarske memorije) pa sve do CD-DVD memorije.
Kes se uvodi jer su savremeni procesori postali znatno brzi od glavnih memorija, pre pristupa glavnoj memoriji procesor pristupa kesu, ako podatak tamo postoji, nazivamo to cache hit, ako ne, cache miss. Najkoriscenije spoljne memorije – usb i eksterni hard diskovi

1.19. Da li je kod na nekom masinskom jeziku prenosiv sa jednog na sve druge racunare? Da li asembler zavisi od masine na kojoj se koristi?

Kod je prenosiv. Da, zavisi od procesora.

1.20. Ukoliko je raspoloziv asemblerski kod nekog programa, da li je moguce jednoznacno konstruisati odgovarajuci masinski kod? Ukoliko je raspoloziv masinski kod nekog programa, da li je moguce jednoznacno konstruisati odgovarajuci asemblerski kod?

Da. Da.

1.21. Koji su osnovni razlozi slojevite organizacije softvera? Sta je sistemski, a sta aplikativni softver?

Zato sto programeri zele da zadrze naredbe na sto apsktraknijem nivou.
Sistemski softver kontrolise hardver i pruza usluge aplikativnom softveru (operativni sistem), a aplikativni softver su aplikacije namenjene korisniku (veb pretrazivaci, video igre, email...)

1.22. Koji su osnovni zadaci operativnog sistema? Sta su sistemski pozivi? Koji su operativni sistemi danas najkorisceniji?

Bolja funkcionalnost softvera. Sistemki softver posreduje izmedju hardvera i aplikativnog softvera koji krajnji korisnici koriste.
Kontrolise i apstrahuje hardver, sinhronizuje rad vise programa, rasporedjuje procesorsko vreme i memorije. Skup funkcija koji programer koristi kako bi postigao zeljenu funkcionalnost hardvera, sakrivajuci pritom konkretne hardverske detalje.
Windows, Linux, iOs, Android...

2.1. Kako su u digitalnom racunaru zapisani svi podaci? Koliko se cifara obicno koristi za njihov zapis?

Svi podaci u digitalnom racunaru su danas zapisani nizovima celih brojeva. Obicno se koriste 2 cifre za zapis podataka, 0 i 1.

2.2. Koje su osnovne prednosti digitalnog zapisa podataka u odnosu na analogni? Koji su osnovni problemi u koriscenju digitalnog zapisa podataka?

Digitalni zapis ne zavisi od toga kakav je kvalitet medijuma na kome je zapisan, dok god su podaci koji su zapisani na tom medijumu razumljivi.
Osnovni problemi digitalnog zapisa su mogucnost da se napravi verna reprezentacija podataka koje zelimo da zapisemo. Takodje moramo imati veoma razvijenu tehnologiju da bi mogli da izvrsimo digitalni zapis, i to je razlog zasto se digitalni zapis jako kasno istorijski javio.

2.3. Sta je Hornerova sema? Opisati Hornerova semu pseudokodom.

Hornerova sema je postupak koji se koristi za jednostavnije izracunavanje neoznacenih brojeva. Efikasniji je od pocetnog postupka jer je u svakom koraku dovoljno izvrsiti samo jedno mnozenje i jedno sabiranje. (n+1 sabiranja i n+1 mnozenja)

(an*an-1.....a1a0)b= (...((an*b+an-1)*b +an-2)...+a1)*b+a0

2.4. Koje su prednosti zapisa u potpunom komplementu u odnosu na zapis u obliku oznacene apsolutne vrednosti?

Znamo uvek koliki je raspon brojeva koje mozemo da zapisemo (-2^{n-1}, $2^{n-1}-1$) i postoji samo jedan izuzetak u argumentu, sam broj -2^{n-1}, nula se zapisuje samo na jedan nacin i vrlo je jednostavno dobiti npr iz pozitivnog negativan broj.

2.5. Sta je to IEEE 754?

IEEE 754 je standard za zapisivanje brojeva u pokretnom zarezu iz 1985. godine, od strane medjunarodne asocijacije Institut inzenjera elektrotehnike i elektronike (Institute of Electrical and Electronics Engineers).

2.6. Sta je to glif, a sta font? Da li je jednoznacna veza izmedu karaktera i glifova?

Elementi pisanog teksta koji najcesce predstavljaju graficke reprezentacije pojedinih karaktera nazivaju se glifovi. Fontovi su skupovi glifova. Ne mora biti jednoznacna.

2.7. Koliko bitova koristi ASCII standard? Sta cini prva 32 karaktera ASCII tabele? Kako se odreduje binarni zapis karaktera koji predstavljaju cifre?

ASCII koristi 7 bitova. Prvih 32 karaktera cine specijalni kontrolni karakteri.
Npr softver koji prikazuje tekst moze vise karaktera predstaviti jednim glifom (tzv ligature), dok jedan isti karakter moze biti predstavljen razlicitim glifovima u zavisnosti od svoje pozicije u reci.
Cifre 0-9 predstavljene su kôdovima $(30)_{16}$ do $(39)_{16}$, tako da se njihov ASCII zapis dobija dodavanjem prefiksa 011 na njihov binarni zapis.

2.8. Navesti barem dve jednobajtne kodne strane koje sadrze cirilicne karaktere.

ISO-8859-5 i Windows-1251.

2.9. Nabrojati bar tri kodne seme u kojima moze da se zapise rec racunarstvo:
Unicode, YUSCII, UTF-8.

2.10. Koliko bitova koristi ASCII tabela karaktera, koliko YUSCII tabela, koliko ISO-8859-1, a koliko osnovna UNICODE ravan? Koliko razlicitih karaktera ima u ovim tabelama?
ASCII - 7 bitova (128 razlicitih karaktera)
YUSCII - 7 bitova (128 razlicitih karaktera)
ISO-8859-1 - 8 bitova (256 razlicitih karaktera)
UNICODE - 16 bitova (2 bajta, 65536 razlicitih karaktera)

2.11. Koja kodiranja teksta je moguce koristiti ukoliko se u okviru istog dokumenta zeli zapisivanje teksta koji sadrzi jedan pasus na srpskom (pisan latinicom), jedan pasus na nemackom i jedan pasus na ruskom (pisan cirilicom)?
Da bi kodirali ovakav tekst moramo da koristimo UNICODE ili UTF-8.

2.12. U cemu je razlika izmedu Unicode i UTF-8 kodiranja?
Unicode sadrzi veliki broj nula koje zauzimaju puno prostora, zbog njih softver za rad u dokumentima u ASCII formatu ne moze da radi bez nekih izmena nad dokumentima kodiranim u UCS-2 standardu.
UTF-8 je algoritam koji svakom dvobajtnom unicode karakteru dodeljuje odredjeni niz bajtova cija duzina varira od 1-3, ovaj algoritam omogucava da se citanjem pocetka samo jednog bajta odredi da li je u pitanju karakter zapisan koriscenjem jednog, dva ili tri bajta.

2.13. Prilikom prikazivanja filma, neki program prikazuje titlove tipa "raeunari æe biti...". Objasniti u cemu je problem.
To znaci da program nije kompatibilan da cita odredjen tip koda u kome je tekst titlova kodiran.

2.14. Sta je to piksel? Sta je to sempl?
Pravugaona matrica komponenti se naziva piksel. Sample je digitalno zapisani signal predstavljen nizom brojeva.

3.1. Po kome je termin algoritam dobio ime?

Po persijskom matematicaru Al-Horezmiju.

3.2. Sta je to algoritam (formalno i neformalno)? Navesti nekoliko formalizma za opisivanje algoritama. Sta tvrdi Cerc-Tjuringova teza? Da li se ona moze dokazati?

Neformalno: Algoritam je precizan opis postupka za resavanje nekog problema u konacnom broju koraka.
Formalno: Algoritam je sve sto mozemo da uradimo u URM programu.
Formalizmi:

- Tjuringove masine
- Rekurzivne funkcije
- Postove masine
- Markovljevi algoritmi
- Masine sa beskonacno mnogo registara (URM)

Cerc-Tjuringova teza: Klasa intuitivno izracunljivih funkcija identicna je sa klasom formalno izracunljivih funkcija, nije moguce dokazati je jer ne mozemo matematicki da definisemo intuitivnost.

3.3. Da li postoji algoritam koji opisuje neku funkciju iz skupa prirodnih brojeva u skup prirodnih brojeva i koji moze da se isprogramira u programskom jeziku C i izvrsi na savremenom racunaru, a ne moze na Tjuringovoj masini?

Ne postoji, sve sto moze da se uradi na Tjuringovoj (ili bilo kojoj drugoj od ovih UR masina) moze i na savremenom racunaru i obratno.

3.4. Da li je svaka urm izracunljiva funkcija intuitivno izracunljiva? Da li je svaka intuitivno izracunljiva funkcija urm izracunljiva?

Svaka URM funkcija jeste intuitivno izracunljiva, jer mi mozemo da pratimo korake masine, da.

3.5. U cemu je kljucna razlika izmedu urm masine i bilo kog stvarnog racunara?

URM masina, kao sto ime naznacuje, ima beskonacan broj registara (Unlimited register machine), dok svaki stvaran racunar ima konacnu memoriju.

3.6. Opisati efekat urm naredbe J(m, n, p).

Instrukcija J je od engleske reci Jump, ona uporedjuje registre pozicija Rm i Rn, ako je m=n (ako su dva registra jednaka po vrednosti), onda sledi skok na komandu broj p u datom programu.

3.7. Da li se nekim urm programom moze izracunati hiljadita cifra broja 2^{1000}?

Moze, moze se svesti na sabiranje tj. u URMu, dodavanje i uz dovoljno vremena, doci do 1000. cifre.

3.8. Da li postoji urm program koji izracunava broj sqrt(2)? Da li postoji urm program koji izracunava n-tu decimalnu cifru broja sqrt(2), gde je n zadati prirodan broj?

Ne moze da izracuna broj sqrt(2), ali moze se izracunati n-ta cifra tog broja, broj 2 dopunimo n nulama, gde n predstavlja n-tu decimal koju zelimo da izracunamo.

3.9. Da li se nekim urm programom moze izracunati hiljadita decimalna cifra broja Pi?
Da, moze, na isti nacin kao iznad.

3.10. Koliko ima racionalnih brojeva? Koliko ima kompleksnih brojeva? Koliko ima razlicitih programa za Tjuringovu masinu? Koliko ima razlicitih programa u programskom jeziku C?
Prebrojivo mnogo. Neprebrojivo mnogo. Prebrojivo mnogo. Prebrojivo mnogo.

3.11. Koliko elemenata ima unija konacno mnogo konacnih skupova? Koliko elemenata ima unija prebrojivo mnogo konacnih skupova? Koliko elemenata ima unija konacno mnogo prebrojivih skupova? Koliko elemenata ima unija prebrojivo mnogo prebrojivih skupova?
Konacno mnogo. Prebrojivo mnogo. Prebrojivo mnogo. Prebrojivo mnogo.

3.12. Koliko ima razlicitih urm programa? Kakva je kardinalnost skupa urm programa u odnosu na kardinalnost skupa prirodnih brojeva? Kakva je kardinalnost skupa urm programa u odnosu na kardinalnost skupa realnih brojeva? Kakva je kardinalnost skupa urm programa u odnosu na kardinalnost skupa programa na jeziku C?
Prebrojivo mnogo. Imaju istu kardinalnost kao prirodni brojevi. Prebrojivi su, ili ima ih prebrojivo mnogo. Nije ista, realnih ima neprebrojivo mnogo dok URM programa im prebrojivo mnogo. Ista je kardinalnost, oba skupa ima prebrojivo mnogo.

3.13. Da li se svakom urm programu moze pridruziti jedinstven prirodan broj (razlicit za svaki program)? Da li se svakom prirodnom broju moze pridruziti jedinstven urm program (razlicit za svaki broj)?
Da, moze, URM programa je prebrojivo mnogo, isto kao i prirodnih brojeva. Da.

3.14. Da li se svakom urm programu moze pridruziti jedinstven realan broj (razlicit za svaki program)? Da li se svakom realnom broju moze pridruziti jedinstven urm program (razlicit za svaki broj)?
Da, URM programa ima prebrojivo mnogo, realnih ima neprebrojivo. Prebrojivom broju mozemo dodeliti neprebrojivo jedinstvenih brojeva. Ne, ne mozemo neprebrojivom broju dodati prebrojivo mnogo jedinstvenih brojeva.

3.15. Kako se naziva problem ispitivanja zaustavljanja programa? Kako glasi halting problem? Da li je on odluciv ili nije? Ko je to dokazao?
Halting problem, on glasi: Ne postoji program koji moze za vrednosti P(x,y) da proveri da li se dati program (x) zaustavlja za ulaznu vrednost y. Nije odluciv. Alan Tjuring je to dokazao 1936. godine.

3.16. Da li postoji algoritam koji za drugi zadati urm program utvrdjuje da li se zaustavlja ili ne? - Ne.

3.17. Da li postoji algoritam koji za drugi zadati urm utvrdjuje da li se zaustavlja posle 100 koraka? - Da.

3.18. Da li je moguce napisati urm program koji za drugi zadati urm program proverava da li radi beskonacno? - Ne.

3.19. Da li je moguce napisati urm program koji za drugi zadati urm program proverava da li vraca vrednost 1? - Ne.

3.20. Na primeru korena uporedite URM sa savremenim asemberlskim jezicima. Da li URM ima neke prednosti?

Savremeni asemblerski jezik moze da uradi isto sto i URM, samo sto je efikasniji po broju instrukcija.

4.1. Ukoliko je raspoloziv kod nekog programa na nekom visem programskom jeziku (na primer, na jeziku C), da li je moguce jednoznacno konstruisati odgovarajuci masinski kod? Ukoliko je raspoloziv masinski kod nekog programa, da li je moguce jednoznacno konstruisati odgovarajuci kod na jeziku C?

Da, moguce je. Da, moguce je.

4.2. Za koji programski jezik su izgradeni prvi interpretator i kompilator i ko je bio njihov autor? Koji su najkorisceniji programski jezici 1960-ih, koji 1970-ih, koji 1980-ih i 1990-ih, a koji danas? Sta je to strukturno, a sta objektno-orijentisano programiranje?

Prvi interpretator i kopilator su napravljeni za jezik FORTRAN. Dzon Bakus.
1960 - COBOL, LISP. 1970 - C, Pascal. 1980 – C++. 1990 - Java, krajem i C#
danas - PHP, JavaScript, Python
Strukturno programiranje predstavlja disciplinovan pristup programiranju, OOP olaksava izradu velikih programa i raspodelu posla unutar vecih programerskih timova.

4.3. Koje su najznacajnije programske paradigme? U koju paradigmu spada programski jezik C? Sta su to proceduralni, a sta su to deklarativni jezici?

Paradigme mogu biti imperativne, objektno orijentisane, funkcionalne, logicke.
C spada u imperativne.
Proceduralni znace da je zadatak programera da opise nacin kojim se dolazi do resenja problema.
Deklarativni zahtevaju od programera da detaljno opise problem, dok se mehanizam programskog jezika onda bavi pronalazenjem resenja problema.

4.4. Sta je rezultat leksicke analize programa? Sta leksicki analizator dobija kao svoj ulaz, a sta vraca kao rezultat svog rada? Sta je zadatak sintaksne analize programa? Sta sintaksni analizator dobija kao svoj ulaz, a sta vraca kao rezultat svog rada?

Rezultat leksicke analize je da je svakoj leksemi(reci koda) pridruzen odredjen token. Leksicki analizator kao ulaz dobija lekseme ili reci a vraca tokene sa recima?
Sintaksni analizator ima zadatak da proveri da li je kod ispravno napisan da bi mogao biti pretvoren u izvrsni kod, u drugim recima da li je kod ispravan.

4.5. Sta karakterise svaki tip podataka? Da li su tipovi prisutni i u prevedenom, masinskom kodu? Sta je to konverzija tipova? Sta znaci da je programski jezik staticki, a sta znaci da je dinamicki tipiziran?

Karakterise ih vrsta podataka koje opisuje, skup operacija koje mogu primeniti nad podacima tog tipa, nacin reprezentacije i detalji implementacije.
Postoje implicitno, ali ne eksplicitno na visem programskom jeziku.
Staticki jezici zahtevaju od programera da definise svaki tip promenljivih i taj tip se ne menja kroz ceo program, dok dinamicki to sam predpostavlja iz zapisa u samom programu, moguce je cak i promena tipova promenljivih u samom programu.

4.6. Koje su osnovne naredbe u programskim jezicima? Sta je to GOTO naredba i po cemu je ona specificna?

Osnovne i najcesce koriscene naredbe su kontrole strukture (if-else-then) i petlje (for, while, do-while, repeat-until).
GOTO naredba je skok naredba na odredjenu poziciju u kodu, specificna je za period pre strukturnog programiranja kada je jako puno koriscena i ljudi su generalno pravili jako nepregledne i neefikasne kodove, po njoj je takodje nastao "spageti efekat".

4.7. Cemu sluze potprogrami? Koji su najcesce vrste potprograma? Cemu sluze parametri potprograma, a cemu sluzi povratna vrednost? Koji su najcesci nacini prenosa parametara?

Potprogrami izoluju odredjena izracunavanja koja se kasnije mogu pozvati na vise razlicitih mesta u razlicitim kontekstima.
Najcesve vrste su funkcije, procedure, sabrutine i metode.
Parametri su ulazne vrednosti koje zadajemo a povratna vrednost sluzi za vracanje rezultata odredjene funkcije, npr ispisivanje na ekranu.

4.8. Sta podrazumeva modularnost programa? Cemu sluze biblioteke? Sta je standardna, a sta rantajm biblioteka?

Modularnost podrazumeva razbijanje veceg programa na nezavisne celine. Celine koje sadrze definicije srodnih podataka i funkcija obicno se nazivaju biblioteke i smestaju se u posebne datoteke.
Biblioteke sluze za visestruku upotrebu pojedinih modula u okviru razlicitih programa.
Standardna biblioteka sadrzi funkcije koje su cesto potrebne programeru. Obicno se staticki povezuju sa programom.
Rantajm biblioteka predstavlja sponu izmedju kompilatora i operativnog sistema, funkcije iz rantajm bibl. se ne ukljucuju u izvrsni kod programa vec se dinamicki pozivaju tokom izvrsavanja programa. (da bi ovo bilo moguce OS mora imati vec instaliranu runtime biblioteku).

4.9. Sta su prednosti, a sta mane mogucnosti pristupa proizvoljnoj memoriji iz programa? Sta su sakupljaci otpada?

Programi sa sakupljacima otpada su obicno sporiji nego bez jer odredjeno vreme odlazi na rad sakupljaca otpada, takodje imamo vecu kontrolu nad memorijom i mozemo da dodeljujemo vrednosti u memoriji samo za ono sto nam je potrebno, ali zbog ovoga postoji i veca sansa za gresku.

4.10. Sta je kompilator, a sta interpretator? Koje su osnovne prednosti, a koje su mane koriscenja kompilatora u odnosu na koriscenje interpretatora?

Kompilatori su programski prevodioci kod kojih su faza prevodjenja i faza izvrsavanja programa potpuno razdvojene. Nakon analize izvornog koda programa viseg programskog jezika kompilatori generisu izvrsni kod i dodatno ga optimizuju, a zatim cuvaju u vidu izvrsnih datoteka. Jedan od problema je sto se prevodjenjem gubi svaka veza izmedju izvrsnog i izvnornog koda programa i najmanja izmena zahteva ponovno prevodjenje programa ili njegovih delova.
Interpretatori su programski prevodioci kod kojih su faza prevodjenja i faza izvrsavanja programa ispripletane. Interpretatori analiziraju deo po deo izvornog koda programa i odmah nakon analize vrse i njegovo izvrsavanje. Obicno se izvrsavaju znatno sporije ali je takodje moguce menjati deo koda i nije potrebno ponovno analiziranje celog koda.

5.1. Kada je nastao programski jezik C? Ko je njegov autor? Za koji operativni sistem se vezuje nastanak programskog jezika C?

C je nastao 1972. i razvio ga je Denis Rici. Vezuje se za UNIX.

5.2. U koju grupu jezika spada C? Za sta se najvise koristi?

C spada u grupu imperativnih programskih jezika. Zbog njegovog direktnog pristupa memoriji i konstrukcije, jednostavno se prevodi na masinski jezik. Najvise se koristio za programe koji su ranije pisani na asemblerskom jeziku.

5.3. Koja je najpoznatija knjiga o jeziku C? Nabrojati sve zvanicne standarde jezika C.

Najpoznatija knjiga o C-u je "The Programing Language C" koju su napisali Kernigen i Rici 1978.
Standardi jezika su: K&R C , ANSI C i ISO C, C90, C99, C11.

5.4. Funkciju kojeg imena mora da sadrzi svaki C program?

Mora da sadrzi Main funkciju.

5.5. Sta su to pretprocesorske direktive? Da li je #include <stdio.h> pretprocesorska direktiva, naredba C-a ili poziv funkcije iz standardne biblioteke?

Pretprocesorska direktiva je posebna naredba u izvornom kodu programa koja se izvršava pre nego se program prevede. Ove direktive se razlikuju od ostalih linija programa: počinju znakom # i završavaju se krajem reda, a ne tačkom i zarezom.
#include <stdio.h> je pretprocesorka direktiva.

5.6. Sta su to datoteke zaglavlja?

To su biblioteke koje sadrze konstante i deklaracije funkcija.

5.7. Sta je standardna biblioteka programskog jezika?

Biblioteka koja sadrzi vec deklarisane funkcije.

5.8. Kojom naredbom se vraca rezultat funkcije u okruzenje iz kojeg je ona pozvana? return 0;

6.1. Koji su osnovni oblici podataka sa kojima se operise u programu?
Promenljive i konstante.

6.2. Sta sve karakterise jedan tip podataka?
Vrsta podataka koje opisuje, nacin reprezentacije, skup operacija koje se mogu primeniti nad podacima tog tipa i broj bitova koji se koriste za reprezentaciju.

6.3. Da li se promenljivoj moze menjati tip u toku izvrsavanja programa?
Ne.

6.4. Da li ime promenjive moze pocinjati simbolom _? Da li se ovaj simbol moze koristiti u okviru imena?
Da. Da.

6.5. Da li ime promenjive moze pocinjati cifrom? Da li se cifre mogu koristiti u okviru imena?
Ne. Da.

6.6. Sta je to inicijalizacija promenljive? U opstem slucaju, ukoliko celobrojna promenljiva nije inicijalizovana, koja je njena pocetna vrednost?
Dodeljivanje vrednosti nekoj promenljivoj.
Njena pocetna vrednost nije odredjena i poziva se neka vrednost koja je postojala pre nje u sistemu.

6.7. Sta je uloga kvalifikatora const?
Da markiramo promenljivu i onemogucimo njeno menjanje, takodje da kazemo drugim programerima da ta promenljiva ne bi trebalo da se menja. Takodje se zove Konstanta.

6.8. Navesti barem jedan tip u jeziku C za koji je standardom definisano koliko bajtova zauzima.
Char.

6.9. Koliko bajtova zauzima podatak tipa char? Ukoliko nije navedeno, da li se podrazumeva da je podatak ovog tipa označen ili neoznačen?
Zauzima tacno 1 bajt. Standard ne propisuje da li je signed ili unsigned, to zavisi od sistema.

6.10. Koja je najmanja, a koja najveća vrednost koju može da ima signed char?
Najmanja vrednost je -128, a najveca 127.

6.11. Koja je najmanja, a koja najveća vrednost koju može da ima unsigned char?
Najmanja vrednost je 0, a najveca 255

6.12. Koja ograničenja važi za dužine u bajtovima tipova short int, int, long int?
Short int zauzima bar 2 bajta, int bar koliko i short int, long int bar koliko i int i bar 4 bajta.

6.13. Ukoliko je tip promenljive int, da li se podrazumeva da je njena vrednost označena ili neoznačena ili to standard ne propisuje?
Podrazumeva se da je signed.

6.14. U kojoj datoteci zaglavlja se nalaze podaci o opsezima celobrojevnih tipova za konkretnu implementaciju?
<limits.h>

6.15. Koja ograničenja važi za dužine u bajtovima tipova float i double?
Nije propisano koliko bajtova zauzimaju, ali double zauzima bar koliko i float.

6.16. U kojoj datoteci zaglavlja se nalaze podaci o opsezima tipova broja u pokretnom zarezu za konkretnu implementaciju?
<float.h>

6.17. Kako se grade izrazi?
Grade se od konstanti, operatora i promenljivih.

6.18. Nabrojati aritmeticke, relacijske, logicke i bitovske operatore.
Aritmeticki: +, - , * , /, %, -, +, ++, - -
Relacijski: ==, !=, >, >=, <, <=
Logicki: !, &&, ||
Bitovski: `, &, |, ^, <<, >>

6.19. Sta je to asocijativnost, a sta prioritet operatora?
Prioritet operatora definise redosled kojim ce se 2 razlicita operatora primeniti na izrazu.
Asocijativnost definise redosled kojim ce se 2 ista operatora izvrsiti.

6.20. Navesti neki operator jezika C koji ima levu i neki operator koji ima desnu asocijativnost.
-, *, /, % ima levu asocijativnost, opetator = ima desnu, i ~ (bitovska negacija)

6.21. Kakav je odnos prioriteta unarnih i binarnih operatora?
Unarni imaju veci prioritet od binarnih.

6.22. Kakav je odnos prioriteta logickih, aritmetickih i relacijskih operatora?
Aritmeticki imaju prioritet u odnosu na relacijske koji imaju prioritet u odnosu na logicke.

6.23. Kakav je odnos prioriteta operatora dodele i ternarnog operatora?
Ternarni imaju veci prioritet u odnosu na dodelu.

6.24. Da li dodele spadaju u grupu nisko ili visoko prioritetnih operatora? Koji operator jezika C ima najnizi prioritet?
Niskog, = ima najnizi prioritet.

6.25. Sta znaci to da se opratori &&, || i ?: izracunavaju lenjo? Navesti primere.

0 || a++, nece doci do a++ jer je ceo izraz odmah zakljucen netacan .
Lenjo izracunavanje je izracunavanje samo neophodnog, sto znaci da se do nekih izraza mozda nece ni doci.

6.26. Sta su to implicitne, a sta eksplicitne konverzije? Na koji nacin se vrednost karakterske promenljive c se moze eksplicitno konvertovati u celobrojnu vrednost?

Implicitne konverzije se vrse automatski kada je potrebno.
Eksplicitne konverzije se vrse na zahtev programera, pretvaraju vrednost promenljive (ne i promenljivu) u drugi tip.

6.27. Sta su to promocije, a sta democije? Koje od narednih konverzija su promocije, a koje democije: (a) int u short, (b) char u float, (c) double u long, (d) long u int?

Promocija predstavlja konverziju vrednosti "manjeg tipa" u vrednost "veceg tipa" (short>int)
democija; promocija ; democija; democija

6.28. Navesti pravila koja se primenjuju prilikom primene aritmetickih operatora.

1) Ako je bar jedan od operanada tipa long double i drugi se promovise u long double;
2) inace, ako je bar jedan od operanada tipa double i drugi se promovise u double;
3) inace, ako je bar jedan od operanada tipa float i drugi se promovise u float;
4) inace, svi operandi tipa char i short se promovisu u int.
5) ako je jedan od operanada tipa long long i drugi se prevodi u long long;
6) inace, ako je jedan od operanada tipa long i drugi se prevodi u long.

U slucaju koriscenja neoznacenih operanada (tj. mesanja oznacenih i neoznacenih operanada), pravila konverzije su nesto komplikovanija:
1) Ako je neoznaceni operand siri11 od oznacenog, oznaceni operand se konvertuje u neoznaceni siri tip;
2) inace, ako je tip oznacenog operanda takav da moze da predstavi sve vrednosti neoznacenog tipa, tada se neoznaceni tip prevodi u siri, oznaceni.
3) inace, oba operanda se konvertuju u neoznaceni tip koji odgovara oznacenom tipu.

6.29. Ako je niz a tipa float inicijalizovan u okviru deklaracije i njegova dimenzija nije navedena, kako se ona moze izracunati? Kada je u deklaraciji niza dozvoljeno izostaviti njegovu dimenziju?

sizeof(a)/sizeof(float) . Ako navedemo inicijalizaciju niza.

6.30. Kada je u deklaraciji niza dozvoljeno izostaviti njegovu dimenziju?

Kada deklarisemo sa inicijalizacijom.

6.31. Da li je nad vrednostima koje su istog tipa strukture dozvoljeno koristiti operator dodele i koje je njegovo dejstvo?

Moze, vrednosti parametara se kopiraju u redosledu u kojem su navedeni.

6.32. Kakva je veza izmedu nabrojivog (enum) i celobrojnog tipa u C-u?
Moguce je deklarisati promenljive, enum se koristi za konkretne brojeve u programu, nikada nije isti.

7.1. Kako se, u okviru naredbe switch, oznacava slucaj koji pokriva sve slucajeve koji nisu vec pokriveni?
default: naredba;

7.2. Ukoliko se u naredbi for (e1; e2; e3) ... izostavi izraz e2, koja ce biti njegova podrazumevana vrednost?
Uvek tacna .

7.3. Da li je u naredbi for(e1; e2; e3) ..., moguce da su izrazi e1 i e3 sacinjeni od vise nezavisnih podizraza i ako jeste, kojim operatorom su povezani ti podizrazi?
Da, zarezom.

8.1. Sta je to deklaracija funkcije, a sta je to definicija funkcije? U kojim slucajevima se koriste deklaracije?
Deklaracija funkcije ukazuje prevodiocu da ce u programu biti koriscena funkcija sa odredjenim tipom povratne vrednosti i parametrima odredjenog tipa. Definicija funkcije navodi deklaracije i naredbe funkcije.
Deklarisanje se koristi kada imamo vise funkcija koje medjusobno pozivaju jedna drugu, da ne bi smo morali da ih redjamo u redosledu i brinemo koja poziva koju, mozemo ih deklarisati na pocetku programa gde ce kompajler samo da procita njihove vrednosti i kasnije da ih definisiemo.

8.2. Sta je to parametar, a sta argument funkcije?
Parametri se navode u definiciji i deklaraciji funkcije, a argumenti se koriste pri pozivu funkcije.

8.3. Koliko najmanje parametara moze da ima funkcija? Kako izgleda deklaracija funkcije koja nema parametara? Koja je razlika izmedu deklaracija void f(void); i void f(); ?
Funkcija ne mora da ima parametre.
void f();
Nema razlike, void f(void); prihvata parametre tipa void, (koji ne mogu da se prime) sto znaci da nema nikakve parametre dok void f() ne prima nikakve parametre.

8.4. Sta se navodi kao tip povratne vrednosti za funkciju koja ne vraca rezultat? Da li takva funkcija mora da sadrzi naredbu return? Da li takva funkcija moze da sadrzi naredbu return? Sta se navodi kao argument return naredbe?
Void. Ne mora. Moze. U funkciji tipa void, ne navodi se nista: return;

8.5. Kako se prenose argumenti funkcija u C-u? Da li se u nekoj situaciji prenos argumenata ne vrsi po vrednosti?
Po vrednosti. Samo po vrednosti, ne.

8.6. Koje su informacije od navedenih, tokom kompilacije, pridruzene imenu niza: (i) adresa pocetka niza; (ii) tip elemenata niza; (iii) broj elemenata niza; (iv) adresa kraja niza?
Adresa pocetka niza, tip elemnata niza .

8.7. Koje se od ovih informacija cuvaju u memoriji tokom izvrsavanja programa?
Adresa pocetka niza .

8.8. Sta se sve od navedenog prenosi kada se ime niza navede kao argument funkcije: (i) adresa pocetka niza; (ii) elementi niza; (iii) podatak o broju elemenata niza; (iv) adresa kraja niza.
Adresa pocetka niza .

8.9. Koja funkcija standardne biblioteke se koristi za izracunavanje duzine niske i u kojoj datoteci zaglavlja je ona deklarisana?
strlen, u string.h

8.10. Cemu je jednako sizeof("abc"), a cemu strlen("abc")?
sizeof= 4 (gleda broj elemenata + terminirajucu nulu), strlen=3 .

8.11. Kako se vrsi prenos unija u funkciju?
Po vrednosti.

9.1. Navesti i objasniti osnovne faze na putu od izvornog do izvrsnog programa.
- Pretprocesiranje – pripremna faza kompilacije; formiranje jedinica prevodjenja
- Kompiliranje - pravljenje objektnih modula od jedinica prevodjenja
- Povezivanje - pravljenje izvrsnog programa od jednog ili vise objektnih modula

9.2. U kom obliku se isporucuje standardna C biblioteka?
U obliku masinskog koda, linkovana staticki.

9.3. Kako se koriscenjem GCC prevodioca moze izvrsiti samo faza pretprocesiranja koda i prikazati rezultat? Kako se moze izvrsiti samo faza kompilacije, bez povezivanja objektnih modula?
gcc -E program.c
gcc -c program.c

9.4. Sta su jedinice prevodenja? Sta su datoteke zaglavlja? U kojoj fazi se od vise datoteka grade jedinice prevodenja?

Jedinice prevodjenja - skup vise datoteka zaglavlja i jednog c programa
Datoteke zaglavlja – zasebne datoteke u koje se idvajaju deklaracije
U pretprocesiranju.

9.5. Sta su objektni moduli? Sta sadrze? Da li se mogu izvrsavati? Kojim procesom nastaju objektni moduli? Kojim procesom se od objektnih modula dobija izvrsni program?

Objektni moduli su rezultat rada kompilatora. Sadrze sve funkcije u masinksom kodu i podatke tj. memorijski prostor rezervisan za promenljive. Ne. Kompilacijom. Povezivanjem.

9.6. Da li svi objektni moduli koji se ukljucuju u kreiranje izvrsnog programa moraju nastati kompilacijom sa programskog jezika C? Da li je moguce kombinovati razlicite programske jezike u izgradnji izvrsnih programa?

Ne. Da.

9.7. Sta znaci da u objektnim modulima pre povezivanja adrese nisu korektno razresene?

Znaci da ne mogu da se pozovu niti izvrsavaju jer jos uvek ne znaju na sta trebaju da pokazuju.

9.8. U kom formatu su objektni moduli i izvrsne datoteke na Linux sistemima?

.o (u ELF formatu)

9.9. Sta je staticko, a sta dinamicko povezivanje?

Staticko povezivanje se vrsi nakon kompilacije, ono u izvrsnu datoteku umece masinski kod svih biblioteckih funkcija.
Dinamicko povezivanje se povezuje prilikom pokretanja programa. Dinamicke biblioteke zajedno sa programom ucitavaju u memoriju racunara i adrese poziva biblioteckih funkcija.

9.10. U kojim fazama prevodenja programa se vrsi prijavljivanje gresaka?

Nakon kompilacije nam kompilator izbacuje greske.

9.11. Kako se koriscenjem GCC prevodioca vrsi odvojena kompilacija i povezivanje programa u datotekama p1.c i p2.c?

gcc -c p1.c
gcc -c p2.c
gcc -o p p1.o p2.o

9.12. Sta je i cemu sluzi program make?

Olaksava generisanje izvrsivih programa od izvornih datoteka.

9.13. Kako se zove alat koji omogucava da se program izvrsava korak-po-korak i da se prate vrednosti promenljivih?

Debugger.

9.14. Cemu sluzi direktiva #include? Gde su sacuvane “ ”, a gde < > direktive?
Pomocu nje se ukljucuje sadrzaj drugih datoteka u datoteku koja se prevodi. " " se nalaze u aktivnom direktorijumu, a < > u sistemskom direktorijumu

9.15. Dokle vazi dejstvo pretprocesorske direktive #define NUM_LINES 1000? Kako se ponistava njeno dejstvo?
Do #undef NUM_LINES ili kraja datoteke . #undef NUM_LINES .

9.16. Kako se od neke tacke u programu ponistava dejstvo pretprocesorske direktive #define x(y) y*y+y?
#undef x(y)

9.17. Kako se moze proveriti da li je neko simbolicko ime definisano u trenutnoj jedinici prevodenja?
#ifdef IME

9.18. Objasniti razliku izmedu makroa i funkcija.
Makroi su samo zamena teksta i ne primaju/prenose argumente. Funkcije sadrze naredbe i argumente, rezervise se prostor u memoriji za njih i vrse se radnje nad njima.

9.19. Kojoj operaciji u tekstualnom editoru je slican efekat direktive #define, a kojoj efekat direktive #include?
find and replace , insert

9.20. Kojim parametrom se GCC navodi da izvrsi samo pretprocesiranje i prikaze rezultat faze pretprocesiranja?
-E

9.21. Koji program pravi od ulaznih datoteka jedinice prevođenja?
Pretprocesor.

9.22. Prilikom generisanja izvršive verzije programa napisanog na jeziku C, koja faza prethodi kom pilaciji, a koja sledi nakon faze kompilacije?
Prethodi predprocesiranje, a sledi povezivanje.

9.23. Kako se zove faza prevođenja programa u kojoj se od objektnih modula kreira izvršivi program? Koji program sprovodi tu fazu?
Povezivanje. Povezivac/linker/uredjivac veza.

9.24. Kako se korišćenjem GCC kompilatora može zasebno prevesti datoteka prva.c, zatim zasebno prevesti datoteka druga.c i na kraju povezati sve pod imenom program?
gcc -c prva.c
gcc -c druga.c
gcc -o program prva.o druga.o

9.25. Koji sve nivoi dosega identifikatora postoje u jeziku C? Koja su dva najčešće korišćena? Kog nivoa je doseg lokalnih, a kog nivoa doseg globalnih promenljivih?

Doseg nivoa datoteke, bloka, funkcije i prototipa funkcije. Najcesce korisceni su doseg nivoa bloka i datoteke.
Doseg lokalnih promenljivih je nivo bloka, funkcije, prototipa funkcije, a nivo datoteke je doseg globalnih promenljivih.

9.26. Nabrojati tri vrste identifikatora prema vrsti povezivanja.

Bez povezanosti, sa unutrasnjom i sa spoljasnjom povezanoscu.

9.27. Koju povezanost imaju lokalne promenljive?

Bez povezanosti su.

9.28. Koji kvalifikator se koristi da bi se onemogućilo da se globalna promenljiva koristi u nekoj drugoj jedinici prevođenja? Koje povezivanje se u tom slučju primenjuje?

static. Unutrasnje povezivanje.

9.29. Koji kvalifikator se koristi da bi se globalna promenljiva deklarisana u drugoj jedinici prevođenja mogla koristiti u drugoj? Koje povezivanje se u tom slucaju primenjuje?

extern. Spoljasnje povezivanje.

9.30. Ako je za neku spoljašnju promenljivu navedeno da je static a za istu tu promenljivu u nekoj drugoj datoteci navedeno da je extern, u kojoj fazi ce biti prijavljena greska?

U fazi povezivanja.

9.31. Koje vrste životnog veka postoje u jeziku C? Koji životni vek imaju lokalne promenljive (ako na njih nije primenjen kvalifikator static)? Koji životni vek imaju globalne promenljive?

Staticke, automatske, dinamicke. Automatski. Staticki.

9.32. Ukoliko je za neku lokalnu promenljivu naveden kvalifikator static, sta ce biti sa njenom vrednoscu nakon kraja izvrsavanja funkcije u kojoj je deklarisana?

Cuva se.

9.33. Opisati ukratko memorijske segmente koji se dodeljuju svakom programu koji se izvrsava.

- *segment koda* – skladisti se izvrsivi kod programa
- *segment podataka* – promenljive koje su zajednicke za ceo program, kao i konstantni podaci
- *steg segment* – svi podaci koji karakterisu izvrsavanje funkcija

9.34. Navesti bar tri vrste podataka koje se cuvaju u svakom stek okviru.
Argumenti funkcije, lokalne promenljive, medjurezultati izracunavanja, adrese povratka...

9.35. Kako se obično obezbeđuje da nema višestrukog uključivanja neke datoteke?
#ifndef #def #endif

10.1. Koje su informacije pridruzene pokazivacu?
Tip i adresa podataka na koji ukazuje.

10.2. Koliko bajtova zauzima podatak tipa unsigned char? Koliko bajtova zauzima podatak tipa unsigned char*?
1 bajt. Zavisi od sistema.

10.3. Ako je velicina koju uzima element nekog tipa t 8 bajtova, koliko onda bajtova zauzima pokazivac na vrednost tipa t?
16 bajtova

10.4. Ako je promenljiva tipa double* na konkretnom sistemu zauzima 4 bajta, koliko bajtova na istom sistemu zauzima promenljiva tipa unsigned char*?
4 bajta

10.5. Na sta se moze primeniti operator referenciranja? Na sta se moze primeniti operator dereferenciranja? Kako se oni oznacavaju? Kakav im je prioritet?
Na promenljive i elemente niza. Na pokazivacku promenljivu. & i *. Imaju visi prioritet od binarnih aritmetickih operatora.

10.6. Kako se oznacava genericki pokazivacki tip? Za sta se on koristi? Da li se pokazivac ovog tipa moze dereferencirati?
Void * . Ukazuje na promenljive razlicitog tipa . Ne, zato sto nije moguće odrediti tip takvog izraza kao ni broj bajtova u memoriji koji predstavljaju njegovu vrednost.

10.7. Kog tipa je promenljiva a, a kog tipa promenljiva b nakon deklaracije short* a, b?
a je short * , b je short

10.8. Kako se prenose argumenti pokazivackog tipa? Sta to znaci?
Po vrednosti. Kopira se originalna vrednost i barata sa tom kopijom.

10.9. Ukoliko je potrebno da funkcija vrati više od jedne vrednosti, kako se to može postići?
Preko argumenata.

10.10. Ako je u okviru funkcije deklarisan niz sa char a[10]; na koji deo memorije ukazuje a+9?
Na deo u kome je sacuvan deseti element .

10.11. Ako je niz deklarisan sa int a[10];, sta je vrednost izraza a? Sta je vrednost izraza a+3? Sta je vrednost izraza *(a+3)?
&a[0]. &a[3]. a[3]

10.12. Kojom komandom se moze zameniti komanda ip=&a[0]?
ip=a

10.13. Da li je vrednost a[i] isto sto i *(a+i)?
Da.

10.14. Ukoliko je niz deklarisan na sledeci nacin: float* x[10], kako se moze dobiti adresa drugog elementa niza?
x + 1

10.15. Neka je niz deklarisan sa int a[10] i neka je p pokazivac tipa int*. Da li je naredba a=p; ispravna? Da li je naredba p=a; ispravna? Koja je vrednost izraza sizeof(p) nakon naredbe p=a;, ako int* zauzima 4 bajta?
Ne. Ne. 4 bajta.

10.16. Niz je deklarisan sa int a[10];. Da li je a+3 ispravan izraz? Da li mu je vrednost ista kao i vrednost a[3] , &a[3] , *a[3] ili a[10]? Da li je *(a+3) ispravan izraz? Da li mu je vrednost ista kao i vrednost a[3] , a[10] , *a[3] ili *a[10] ?
Ne. &a[3]. Da. a[3] .

10.17. Ako je kao argument funkcije navedeno ime niza, sta se sve prenosi u funkciju?
Adresa pocetka niza

10.18. Koji prototip je ekvivalentan prototipu int f(char s[])? Da li ima razlike izmedju prototipova void f(char *x); i void f(char x[]); ? Da li ima razlike između deklaracija char* x; i char x[]; Da li su obe ispravne?
int f(char *s). Nema razlike. Nema razlike. Ispravne su obe.

10.19. Ako su promenljive p i q tipa double*, za koliko ce se razlikovati vrednost p i q nakon komande p=q+n?
p - q = n * sizeof(double)

10.20. Da li postoji razlika izmedju komandi (*p)++ i *(p++)?
Da.

10.21. Za koliko bajtova se menja vrednost pokazivaca p tipa int* nakon naredbe p+=2;? Ako je p tipa float*, za koliko se razlikuju vrednosti p+5 i p? Za koliko bajtova se menja vrednost pokazivaca tipa T*, nakon naredbe p+=2;? Ako je p pokazivac tipa T, cemu je jednako p+3?
p = p+2*sizeof(int)
za 5*sizeof(float)
2*sizeof(T) , p+3*sizeof(T)

10.22. U kom segmentu memorije se tokom izvrsavanja programa cuvaju konstantne niske?

U segmentu podataka

10.23. Nakon koda void f(){ char* a=''Zdravo''; char b[]=''Zdravo''; …} u kom segmentu memorije je promenljiva a? U kom segmentu je ono na sta pokazuje promenljiva a? U kom segmentu su elementi niza b? Da li se vrednosti a i b mogu menjati? Koliko bajtova zauzima a, a koliko p?

U stek segmentu. U segmentu podataka. U segmentu podataka. a se moze menjati, b ne moze. Za a zavisi od operativnog sistema, b zauzima 7

10.24. Koliko bajtova zauzima I u kom delu memorije niz koji je u okviru neke funkcije deklarisan sa char* a[10];?

Zavisi od OS-a. U stek segmentu.