**1. U kakvoj su zavisnosti razumljivost programa i broj instrukcija bezuslovnog skoka?**

**2. Teorema o strukturiranom programiranju glasi**:

Svaki programski zadatak je moguće logički ostvariti primjenom samo elementarnih struktura sekvence, izbora i ponavljanja. Primjena ove teoreme nas dovodi do potrebe utvrđivanja metodologije razvoja programskih struktura. Najčešće se koristi top-down metoda.

**3. Koje su osnovne programske strukture?**

Osnovne programske strukture su:

* Sekvenca (slijed)
* Selekcija (izbor)
* Iteracija (ponavljanje)

**4. Šta je to sekvenca?**

Sekvenca predstavlja niz instrukcija obrade, koje se izvode uzastopno

**5. Šta je to selekcija (grananje)?**

Selekcijom biramo jednu od ponuđenih sekvenci za izvršavanje, i to prema vrijednosti logičkog (uvjetnog) izraza.

**6. Šta je to iteracija (petlja, ciklus)?**

Iteracijom određujemo da određene sekvencu treba izvršiti više puta

**7. Koliko ulazaka a koliko izlazaka ima u/iz osnovne programske strukture?**

**8. Koliko uskakanja a koliko iskakanja ima u/iz osnovne programske strukture?**

**9. Nabrojati vrste upravljačkih struktura**.

**10. U strukturu selekcije (granjanja) spadaju:**

Sekvenca se implementira korišćenjem bloka u kome se nalazi jedna ili više linija koda. Primjer:

Početak sekvence označava se lijevom vitičastom zagradom **{** , a kraj sekvence desnom vitičastom zagradom **}**. Svaka instrukcija obrade unutar sekvence završava znakom točka-zarez **;**

**11. U petlje (iteracije, cikluse) spadaju:**

for, foreach, while, do-while

**12. U pogledu mjesta izlaska iz njih petlje (iteracije, ciklusi) se dijele na**

**13. Grafički prikazati *if* strukturu i opisati način njenog rada ukoliko se u njenom sastavu nalazi jedna naredba**

****

Naredbu IF koristimo onda kada želimo izvršiti neki kod samo ako je vrijednost nekog odnosnog izraza istinita.

**14. Napisati obrazac za *if* strukturu i opisati način njenog rada ukoliko se u njenom sastavu nalazi jedna naredba.**

if (izraz)   
naredba;

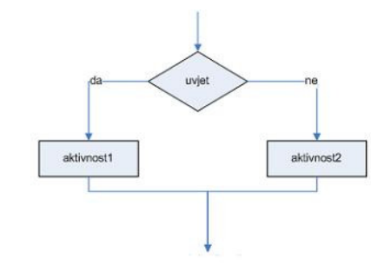
Naredbu IF koristimo onda kada želimo izvršiti neki kod samo ako je vrijednost nekog odnosnog izraza istinita.

**15. Grafički prikazati *if* strukturu i opisati način njenog rada ukoliko se u njenom sastavu nalazi više naredbi.**

**16. Napisati obrazac za if strukturu i opisati način njenog rada ukoliko se u njenom sastavu nalazi više naredbi**

**17. Šta su to ugnježdene (obuhvaćene) if strukture?**

**18. Grafički prikazati if-else strukturu i opisati način njenog rada**

****

Ako bi smo htjeli napraviti da ukoliko uvjet nije zadovoljen da se izvrši neki drugi kod onda koristimo If-Else naredbu.

**19. Napisati obrazac za if-else strukturu i opisati način njenog rada**

if(godine>18) {

cout << "Vi ste punoljetni!"; }

else {

cout << "Vi ste maloljetni!"; }

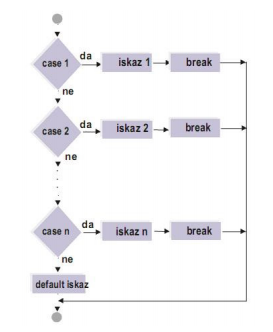
Ako bi smo htjeli napraviti da ukoliko uvjet nije zadovoljen da se izvrši neki drugi kod onda koristimo If-Else naredbu.

**20. Grafički prikazati ugnježdene (obuhvaćene) if-else strukture i opisati način njihovog rada**

**21. Objasniti pojam programske strukture skretnica (switch).**

Ova naredba sluzi za provjeravanje više uvjeta od jednom i izvršavanja onoliko radnji koliko je uslova zadovoljeno odnosno tačno**.**

**22. Grafički prikazati switch strukturu i opisati način njenog rada**

****

Ova naredba sluzi za provjeravanje više uvjeta od jednom i izvršavanja onoliko radnji koliko je uslova zadovoljeno odnosno tačno**.**

**23. Napisati obrazac za switch strukturu i opisati način njenog rada**.

switch (varijabla){

case mogucnost1:

naredba;

break;

case mogucnost2:

naredba;

break;

...

}

Ova naredba sluzi za provjeravanje više uvjeta od jednom i izvršavanja onoliko radnji koliko je uslova zadovoljeno odnosno tačno**.**

**24. Objasniti pojam programske naredbe break.**

Ovom rijecju se zatvara niz naredbi i daje se mogucnost novom uslovu.

**25. Definisati (odrediti) pojam programske strukture petlja (ciklus, iteracija**).

Iteracijom (petljom) određujemo da određene sekvence treba izvršiti više puta

**26. Koji su kriterijumi za napuštanje petlje?**

Kriteriji za napuštanje petlje su uslov koji smo postavili na samom početku petlje , tj izvršavaju se naredbe sve dok uslov ne bude zadovoljen.

**27. Sa kojom vrijednošću indeksa petlje se izlazi iz petlje?**

**28. Koliko puta se izvršava petlja ukoliko su parametri petlje integer-i?**

**29. Koliko puta se izvršava petlja ukoliko su parametri petlje real-ovi?**

**30. Koji su najpoznatiji numerički metodi nalaženja nule funkcije?**

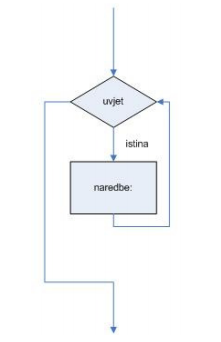
**31. Grafički prikazati petlju sa izlaskom na dnu i opisati način njenog rada.**

**32. Grafički prikazati petlju sa izlaskom na vrhu i opisati način njenog rada.**

**33. Šta su to ugnježdene (obuhvaćene) petlje?**

Ugnježdene petlje su petlje koje se pišu jedna unutar druge.

**34. Grafički prikazati while strukturu i opisati način njenog rada.**



Naredba while ili while petlja koristi se za ponavljanje segmenata koda kod kojeg broj ponavljanja nije unaprijed poznat.

**35. Napisati obrazac za while strukturu i opisati način njenog rada.**

while (uslov\_izvodjenja)

{

//blok\_naredbi

}

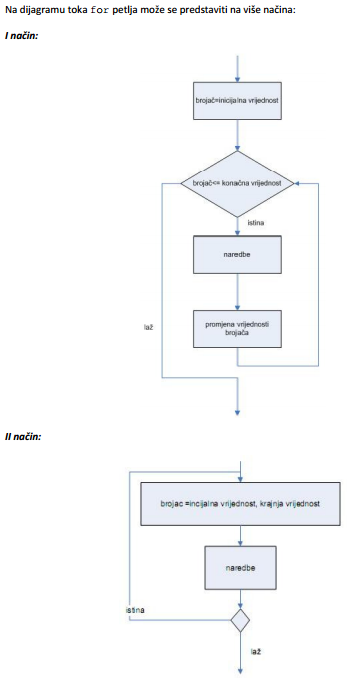
Naredba while ili while petlja koristi se za ponavljanje segmenata koda kod kojeg broj ponavljanja nije unaprijed poznat.

**36. Objasniti pojam programske naredbe continue**

Naredba continue ima za zadatak da prekine samo trenutačnu iteraciju petlje u kojoj se nalazi.

**37. Grafički prikazati for strukturu i opisati način njenog rada.**

Često u programima trebamo ponavljati dijelove koda. Ako je broj ponavljanja poznat prije ulaska u petlju, najprikladnije je koristiti for petlju

****

**38. Napisati obrazac za for strukturu i opisati način njenog rada**

for ( pocetni\_izraz; uslov\_izvodjenja; izraz\_prirasta )

{

//blok\_naredbi

}

Često u programima trebamo ponavljati dijelove koda. Ako je broj ponavljanja poznat prije ulaska u petlju, najprikladnije je koristiti for petlju

**39. Načiniti primjer za for strukturu i opisati način njenog rada.**

for ( i=1 , i <=10,i++)

{

Cout << „1“ <<endl;

}

For petlja nam omogućava da vršimo operacije i ispišemo onoliko puta koliko to želimo. U ovom slučaju smo broj 1 ispisali 10 puta, postavili smo brojač koji broj od 1 do 10 za 1+.

**40. Šta su to inkrementalni operatori u jeziku C++?**

Ikrementalni opratori (++) omogućavaju prigodan način za povećavanje brojne varijable za 1.

**41. Šta su to dekrementalni operatori u jeziku C++?**

Dekrementalni operatori (--) omogućavaju prigodan način za smanjivanje brojne varijable za 1.

**42. Šta su to prefiksni operatori u jeziku C++?**

Prefiksni operatori u c++ jeziku su operatori koji se nalaze ispred varijable.

**43. Šta su to sufiksni operatori u jeziku C++?**

Prefiksni operatori u c++ jeziku su operatori koji se nalaze iza varijable.

**44. Šta je to program?**

Računarski program, softverski program ili jednostavno program jeste spisak naredbi napisan u nekom programskom jeziku namijenjen za određenu računarsku platformu. Računarski programi napisani u nekom programskom jeziku su razumljivi čovjeku, ali te naredbe ne razumije i računar, zbog toga se kod računarskog programa pomoću kompajlera mora prevesti u binarni kod da bi ga računar razumio i uopće izvršavao

**45. Koja su dva logička dijela instrukcije?**

Operacioni i adresni

**46. Šta definiše (određuje) operacioni dio instrukcije?**

**47. Od čega zavisi sadržaj adresnog dijela instrukcije?**

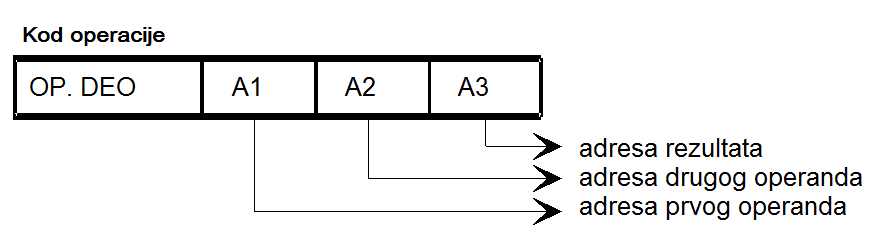
**48. Vrste instrukcija:**

Jednoadresna, dvoadresna, troadresna

**49. Koja je struktura instrukcije za obradu podataka?**

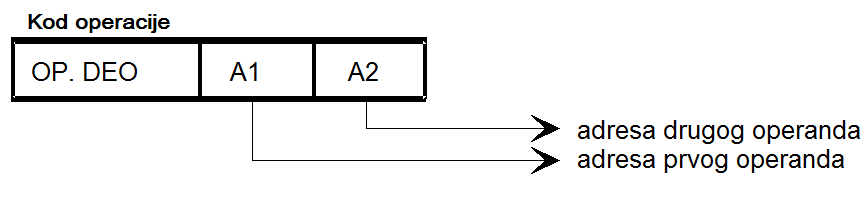
**50. Koliko najčešće adresa sadži instrukcija za obradu podataka?**

**51. Grafički prikazati troadresnu instrukciju za obradu podataka i opisati način njenog rada.**

****

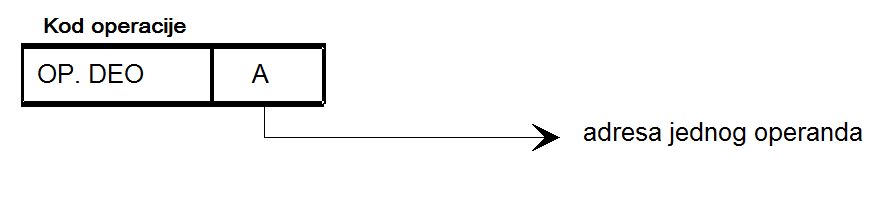
Kod ovog formata, instrukcija je podeljena na četri dela. Prvi dio predstavlja kod instrukcije. Na osnovu koda instrukcije,procesoru se zadaje tip obarde izvršavanja tekuće instrukcije. Drugi deo instrukcije označen sa A1 predstavlja predstavlja adresu prvog izvorišnog operanda. Drugi deo insturkcije označen sa A2 predstavlja adresu drugog izvorišnog operanda a treći dio označen sa A3 predstavlja niz bitova koji specifiraju adresu na kojoj će se smestiti rezultat opreacije instrukcije.

**52. Grafički prikazati dvoadresnu instrukciju za obradu podataka i opisati način njenog rada.**

****

Prvi dio predstavlja kod instrukcije. Drugi deo A1 predstavlja u isto vreme adresu prvog izvorišnog operanda i adresu odredišnog operanda. Treći dio predstavlja adresu drugog izvršnog operanda.

**53. Grafički prikazati jednoadresnu instrukciju za obradu podataka i opisati način njenog rada.**

****

Za razliku od dvoadresnog formata, kod jednoadresnog formata unutar same instrukcije nemamo drugi izvrorišni operand. U ovom slučaju, drugi izvorišni operand predstavlja sardžaj akumulatora.

**54. Koja je struktura upravljačke instrukcije?**

**55. Koliko adresa najčešće sadži upravljačka instrukcija?**

**56. Šta sadži registar: pokazivač instrukcija?**

**57. Šta sadži registar instrukcija?**

**58. Koji je zadatak Aritmetičko - Logičke jedinice?**

ALU predstavlja višefunkcionalnu kombinacionu mrežu koja može da obavlja različite aritmetičke i logičke operacije nad dva n-bitska binarna broja. Аritmеtičkо-lоgičkа јеdinicа је lоgičkо kоlо kоје izvоdi аritmеtičkе (sаbirаnjе, оduzimаnjе, mnоžеnjе i djеlјеnjе) i lоgičkе оpеrаciје (upоrеđivаnjе dvе vrijеdnоsti dа bi sе оdrеdilа vеćа, оdrеđivаnjе dа li је iskаz istinit ili nе...).

**59. Šta je to akumulator i koji je njegov zadatak?**

Akumulator je poseban registar koji služi za prihvatanje operanada i rezultata operacije u ALU.

**60. Skicirati strukturu centralnog procesora i memorije hipotetičkog jednoadresnog računara.**



**61. Načiniti program u opisnom obliku za sabiranje (zbrajanje) dva broja koji se nalaze na lokacijama 1 i 2 i rezultat smjestiti na lokaciju 3.**

**62. Načiniti program u binarnom obliku za sabiranje (zbrajanje) dva broja koji se nalaze na lokacijama 1 i 2 i rezultat smjestiti na lokaciju 3.**

**63. Načiniti program u simboličkom obliku za sabiranje (zbrajanje) dva broja koje se nalaze na lokacijama 1 i 2 i rezultat smjestiti na lokaciju 3.**

**64. Načiniti program u višem programskom jeziku za sabiranje (zbrajanje) dva broja koji se nalaze na lokacijama 1 i 2 i rezultat smjestiti na lokaciju 3.**

**65. Koliko bita zauzima operacioni kôd, a koliko adresni kôd ukoliko se memorija sastoji od 16 ćelija od po jedan bajt?**

**66. Koliko bita zauzima operacioni kôd, a koliko adresni kôd ukoliko se memorija sastoji od 61 ćelije od po jedan bajt?**

**69. Prevođenje programa (nacrtati sliku)**

**70. Tabela lokacija (nacrtati sliku).**

**71. Povezivanje i punjenje programa (nacrtati sliku).**

**72. Šta je to biblioteka gotovih programa?**

**73. Ko čini biblioteku gotovih programa?**

**74. Navesti bar dvije biblioteke gotovih programa u jeziku C++.**