1. Što je naredba?

U najužem smislu naredba je neka kombinacija električnih napona koja dovedena na odgovarajuće stezaljke središnje jedinice za obradu izaziva izvršenje željene radnje.

2. Navedite barem tri oblika prikaza naredbe.

- Binarni: 10100101
- Heksadekadski: A5
- Mnemonički: SUM A,B
- Viši oblik naredbe: Write('Hura');

3. Navedite razliku između sintaksne i logičke pogreške u programu.

Sintaksna pogreška je pogreška pri pisanju programa (nije sve po zadanim pravilima) i program se ne može izvršiti, a logička pogreška je pogreška s kojom se može izvršiti program, ali rezultat neće biti točan.

Matematika: $P = r2\pi$

C - sa sint. i log. pogreškom: P = r + r * 3.14

C - s logičkom pogreškom: P = r + r * 3.14;

C - ispravno: P = r * r * 3.14;

4. Navedite šest koraka u razvoju programa.

1. Analiza zadatka

2. Dijagram toka

3. Definiranje ulaza i izlaza

4. Pisanje programa

5. Testiranje programa

6. Izrada dokumentacije

5. Navedite prednosti i nedostatke programa pisanih u asembleru i u višem programskom jeziku.

PREDNOSTI: jednostavnost pisanja programa, njihova razumljivost, prenosivost.

NEDOSTACI: relativno sporije izvođenje, nemogućnost izravnog nadzora sklopovskih dijelova računala.

6. Napišite što je prevoditelj i navedite primjer?

Primjer : C++, Basic, Fortran, Cobol, Clipper

Prevoditelj je program koji služi za prevođenje izvornog programa u strojni jezik

7. Napišite što je interpreter i navedite primjer?

Interpreter je program koji izvršava druge programe tj, interpreter je jezični prevoditelj koji svaku naredbu izvornog programa u trenutku izvođenja program se prevodi u binarni oblik strojnog jezika. Primjeri: Lotus, dBase, GW Basic.

8. Objasnite što je strukturno programiranje.

Strukturno programiran je skup je programskih metoda (modularno programiranje, deklariranje svih varijabli prije korištenja, izbjegavanje neki naredbi nestrukturnog programiranja (goto)), koje vode k logičkoj organizaciji i čitljivosti programa. Logička organizacija olakšava pisanje, održavanje i popravak programa. Tjeraju programere na programsku disciplinu i urednost, ograničavaju slobodu programera.

9. Objasnite što je objektno orijentirano programiranje.

Objektno orijentirano programiranje omogućuje veću kvalitetu i pouzdanost, ali povećava proizvodnost programera. Ono nam omogućuje da objekt ne mora pisati točno u programu u kojem je programiramo, nego ga možemo napisati u bilo kojem programu i onda ga samo prenijeti u pravopisnom obliku u program u kojem programiramo. Objekt je zaseban program koji je sposoban odraditi neku određenu zadaću. Moguće je povezivati više takvih različitih objekata u jedan program.

10. Objasnite namjenu i navedite primjere programskih jezika neovisnih o sklopovlju i operacijskom sustavu.

Jednom napisani i prevedeni program u načelu se može izvršavati na bilo kojem računalu, ali u praksi to nije bilo moguće (strojni jezik koji je rezultat prevođenja ne može se izvršavati na različitim vrstama računala) pa se potaknuo razvoj novih programskih jezika za pisanje znakovima koje prihvaća svako računalo. Primjeri: HTML, VRLM, Java.

11. Ukratko objasnite strukturu C programa.

- C je programski jezik slobodnog formata, nema pravila što se tiče stila pisanja. Naredbe mogu početi u bilo kojem stupcu reda, te se mogu umetnuti i prazni redovi za odvajanja pojedinih cjelina programa. U istom redu moguće je napisati nekoliko naredbi odvojenih točka-zarezom, ključne riječi i naredbe se pišu malim slovom,prepoznavanje velikih i malih slova.

- C programi moraju sadržavati linije, u svakom programu mora postojati jedna glavna funkcija – **main**, naredba iza prve otvorene vitičaste zagrade u main funkciji je prva naredba koja se izvodi.Dio programa između vitičastih zagrada { }naziva se blok, jedan blok se može nalaziti unutar drugog bloka, svaka naredba završava s **;**

12. Navedite barem 5 ključnih riječi C jezika prema ANSI standardu.

break, int, double, float, char, else, return, if, for

13. Objasnite postupak prevođenja i pokretanja programa C programa.

Prevođenje i pokretanje programa ovisi o operacijskom sustavu i prevoditelju. Npr.na UNIX-u program se može prevesti naredbom cc prvi.c a kao rezultat uspješnog prevođenja dobiti ćemo datoteku a.oat. U MS-Dosu na PC računalima: za Microsoft C prevoditelj: cl prvi.c, za Borland C++: bcc prvi.c ili Make program... i kao rezultat dobije se izvrsna datoteka prvi.exe. Moderni prevoditelji podržavaju integrirano razvojno okruženje unutar kojeg se pišu, uređuju, ispravljaju, testiraju i izvode programi: Visual Studio: Build.

14. Navedite ulogu i primjer korištenja pretprocesorskih naredbi.

Uloga pretprocesorskih naredbi je da se izvrši transformacija programa prije izvođenja, recimo da se uključi standardno zaglavlje koje sadrži definiciju tipova i funkcijske prototipove. Osnovna svrha im je parametrizacija programa.

Pr.: #include<stdio.h> (nalaže prevoditelju da u program uključi standardno zaglavlje koje sadrži definiciju tipova i funkcijske prototipove.)

15. Objasnite korištenje uvjetnog uključivanja pretprocesorskih naredbi.

Objašnjeno u prethodnom zadatku.

16. Navedite osnovne tipove podataka u C-u i za svaki tip po jedna primjer.

char (znakovni tip) a,b,c,d,...

int (cjelobrojni tip) 10,15,73,...

float (realni tip) 1.24,3.14, 56.987,...

double (realni tip u dvostrukoj preciznosti) 14767998324.932,...

17. Ukratko objasniti i na primjeru pokazati ispravno dodjeljivanje imena varijablama u C-u.

Dodjela imena vrši se prema slijedećim pravilima:

- imena su sastavljena od slova i brojki
- prvo slovo mora biti znak ili znak podcrtavanja (_) preporučljivo koristiti samo za zamjenu praznine u imenu,ali ne i na početku imena
- velika i mala slova se razlikuju
- duljina imena varijabli je proizvoljna
- ključne riječi (for, if, else, while,...) su rezervirane i ne mogu se koristiti kao imena varijabli

Pr.: suma, površina, x 1, sumznam 1.

18. Objasniti značenje osnovnih tipova podataka, te bitnije prefikse i kvalifikatore koji se s njima kombiniraju.

Osnovni tipovi podataka nam dozvoljavaju rad s određenom vrstom znakova (s određenim formatom) i omogućuju ispisivanje nekog znaka kao rezultata izvršenja programa, a prefiksi i kvalifikacije nam omogućuju njihove „nadogradnje", tj. njihovo preuređenje.

Osnovni tipovi:

int (cjelobrojni tip)

float (realni tip)

char (znakovni tip)

double (realni tip dvostruke preciznosti) Prefiksi i kvalifikatori:

long (povećava raspon vrijednosti koje varijabla može sadržavati) short (smanjuje raspon vrijednosti koje varijabla može sadržavati) signed (pridruživanje samo pozitivnih vrijednosti) unsigned (dozvoljava pridruživanje i pozitivnih i negativnih vrijednosti)

19. Poredajte od najmanjeg prema najvećem char, double, float, short i long tipove podataka.

char, short, int, long, float, double

20. Na barem dva primjera pokažite prikaz znakovnih konstanti u C-u.

- uključenje prijelaza u novi red:

printf (“Pritisnite enter za nastavak\n”);

- definiranje simboličke konstante za escape sekvencu:

#define ZVONO ‘\a’ /* zvucni signal terminala (ASCII kod 7) */

21. Navedite i objasnite barem tri escape sekvence u C-u.

\n (prelazak u novi red)

\a (alarm(zvučni signal))

\r (postavljanje kursora na početak retka)

\t (tabulator)

\v (vertikalni tabulator)

22. Ukratko objasnite i na primjeru pokažite predstavljanje stringa.

String konstanta je predstavljena dvostrukim znakovima navoda unutar kojih se može naći bilo koji znak ili grupa njih. Ovaj tip znakova nema ključnu riječ, već se predstavlja poljem znakova, ako sadrži samo znamenke ne radi se o broju pa se ne mogu s njim vršiti matematičke operacije, ispis se postiže naredbom printf, sve string konstante završavaju nul znakom, duljina je određena brojem znakova do nul znaka.

Pr.: "predstavljanje stringa"

za ispis: printf ("predstavljanje stringa");

23. Ukratko objasnite i na primjeru pokažite prioritet operatora u C-u.

Prioritet matematičkih operatora se dijeli na dvije grupe:

1. množenje, dijeljenje, modul (ostatak cjelobrojnog dijeljenja)

2. zbrajanje, oduzimanje.

Ako mat.izraz sadrži više operatora istog prioriteta C izračunava izraz slijeva na desno, a pri korištenju zagrada prioritet ima izraz unutar zagrade,a ostali dio izračunava se dalje normalno po određenim prioritetima.

Pr.: 4+4*8=4+32=36

(4+4)*8=8*8=64

4*4/2%3=16/2%3=8%3=2

24. Navedite i ukratko objasnite relacijske operatore u C-u.

Relacijski operatori:

= = (jednako) >(veće)

< (manje)

>= (veće ili jednako)

<= (manje ili jednako)

!= (različito)

Relacijski operatori uspoređuje dva podatka (odnos između njih(veći od ,manji od, jednako)) i pojavljuju se između dviju konstanti, varijabli ili izraza. Relacijski izraz daje istinitu (razl. od 0) ili lažnu (0) vrijednost. Koristimo ih kod postavljanja uvjeta u raznim petljama.

25. Navedite i ukratko objasnite logičke operatore u C-u.

Logički operatori: && (logički operator AND) || (logički operator OR) ! (logički operator NO)

Logički operatori nam služe za ispitivanje više od jednog skupa varijabli. Najveći prioritet ima NO, a zatim AND, te na kraju OR.

26. Navedite i ukratko objasnite bit operatore u C-u.

Primjenjuju se na interne reprezentacije podataka u memoriji,a ne samo na vrijednosti varijable. Pomoću njih C dozvoljava programiranje na nižoj razini, za razliku od drugih viših programskih jezika. C sadrži 4 bit operatora i svaki izvodi bit po bit operaciju s internim podatcima. Mogu se primjenjivati jedina s char, int i long tipovima podataka.

Bit operatori:

& (bit I)

| (bit ILI)

^(bit ekskluzivno ILI)

~ (bit jednostruki komplement)

27. Pokažite na primjerima matematičke operatore koje podržava C.

Pr.: 4+4*8=4+32=36

(4+4)*8=8*8=64

4*4/2%3=16/2%3=8%3=2

28. Pokažite na primjeru kako se u C-u prikazuje niz znakova.

```
#include <stdio.h>
main()
{
printf ("niz znakova");
}
```

29. Objasnite na primjeru korištenje if-else naredbi.

```
#include<stdio.h>
#include<conio.h>
```

```
main()
{
 float n;
 printf("Upisite jedan broj: ");
 scanf("%f",&n);
 if (n<0);
   printf("broj %.2f je negativan ",n)
 else
   printf("broj %.2f je pozitivan",n);
 getch ();
}
```

Provjerava je li uneseni broj pozitivan ili negativan.

30. Objasnite i na primjeru pokažite razliku između for, while i do-while petlji.

Razlika između for, while i do while je u tome što se kod naredbe for program izvršava određen broj puta (njega postavljamo uvjetom u naredbi), naredba while se ponavlja u beskonačnost dok god je uvjet zadan na početku zadovoljen,a ako nije izvodi se prva naredba iza petlje, a naredba do while se ispituje uvjet tek na kraju i ako je uvjet ispunjen nastavlja se izvođenje petlje u beskonačnost; u suprotnom program prestaje s izvođenjem petlje i ide na prve slijedeću naredbu. Razlika naspram while petlje je u tome što se tijelo do-while petlje uvijek izvršava barem jednom.

31. Objasnite i na primjeru pokažite upotrebe switch naredbe.

Najpogodnija naredba za ostvarenje višeznačne odluke na temelju konstantnih cjelobrojnih ili znakovnih vrijednosti. Ako vrijednost izraza odgovara nekom od konstantnih izraza vrši se grananje na pripadajući blok naredbi koji obično završava naredbom break za trenutni izlaz iz switch petlje. Posljednji element je default i predstavlja slučaj koji se izvodi ako nijedan od prethodnih slučajeva nije ispunjen i on se može izostaviti.

```
#include<stdio.h>

#include<conio.h>

main()

{

int a;

printf("Upisite ocjenu: ");

scanf("%d",&a);

switch(a)

{

case 1: printf("nedovoljan");break;

case 2: printf("dovoljan");break;

case 3: printf("dobar");break;

case 4: printf("vrlo dobar");break;

case 5: printf("odlican");break;

default: printf("Kriva ocjena!");break;

}

getch ();

}
```

32. Navedite ulogu funkcija u C-u, te pokažite kako se funkcije deklariraju.

Funkcije su osnovno sredstvo u C-u kojim se programi raščlanjuju na manje programske cjeline. One izvršavaju neki specifični zadatak, a mogu se koristiti i u drugim programima umjesto da se počinje ispočetka. Funkcija main je glavna, izvršava se prva i poziva druge funkcije.

```
int getmax (int V[10])

{

int i,max; max=V[0]; for(i=0;i<10;i++) {

if(max<V[i])max=V[i];

}

return max;

}
```

33. Ukratko navedite pravila pisanja funkcija.

Svaka funkcija mora imati ime; pravila za funkcijsko ime(kao varijabla): može imati do 31 znaka,početi sa slovom ili _ i sadržavati slova,brojeve i _; iza funkcijskog imena dolazi par zagrada, a unutar njih se može nailaziti skup parametara odvojenih zarezima; tijelo funkcije koje počinje iza zatvorene desne zagrade mora se nalaziti unutar vitičastih zagrada{}; main poziva drugu funkciju; u C-u ne postoji ograničenje pozivanja funkcija pa svaka f može pozvati bilo koju drugu; prilikom prijenosa podataka iz jedne f u drugu lokalna varijabla se mora staviti u zagrade u pozivajućoj i pozvanoj f; prilikom prijenosa argumenata u f pozvana f ne zna tip podatka prenesene varijable pa ispred imena parametara treba navesti tip.

34. Navedite primjer deklariranja funkcije.

Deklaracija:

<povratni_tip> ime_funkcije (<tip> arg1, <tip> arg2);

Primjer deklaracije funkcije:

float kvadrat(**float**);

35. Što je funkcijski prototip?

-prototip funkcije predstavlja njezin model kojim se informira C prevoditelj o tipovima funkcijskih parametara i povratne vrijednosti. Oni sprječavaju mogućnost pogreške pri prijenosu podataka neodgovarajućeg tipa u funkciju. Identičan je funkcijskoj definiciji samo ima ; na kraju.

36. Objasnite na primjeru razliku između lokalnih i globalnih varijabli.

Lokalne varijable su one koje se nalaze iza lijeve otvorene zagrade blok naredbi (vidljive samo u bloku),a globalne se nalaze izvan funkcije, prije funkcijskog imena (vidljive u svim funkcijama). Globalne su pogodnije u slučaju da se veliki broj varijabli mora dijeliti između funkcija(izbjegavamo velik broj argumenata); postojane su i zadržavaju vrijednost od jednog do drugog poziva funkcije, a lokalne su definirane u jednoj funkciji i traju dok se funkcija izvodi.

37. Koja naredbe omogućuje organiziranje programa u više datoteka? Pokazati primjer korištenja.

Naredba extern.

38. Što su automatske, a što statičke varijable i kako se deklariraju?

Automatske varijable su sve lokalne varijable (po završetku f u cijelosti se brišu), za deklaraciju je ključna riječ **auto**, ali ona se rijetko koristi.

Pr. deklaracije: auto int i; auto double x;

Statičke varijable su sve globalne varijable, dok se lokalne statičke varijable posebno deklariraju pomoću ključne riječi **static**. Lokalne statičke varijable se ne brišu po završetku funkcije, nego zadržavaju svoju vrijednost.

Pr.: miscO {int i; static int j=5,k;}

39. Što je pokazivač i kako se deklarira? Pokazati i na primjeru.

Pokazivači su varijable koje sadrže adrese drugih varijabli ili vrijednosti. Pokazivač sadrži adrese obične podatkovne varijable, tj. pokazivač pokazuje na podatke,a ne sadrži ih.

Deklaracija: sastavni dio deklaracije je tip podatka na koji pokazuje. Mogu se deklarirati kao lokalni i globalni. Kod deklaracija se koriste operator * - deferentni operator(pokazivač pokazuje samo na vrij. naznačenog tipa) i &.

Pr.:

```
#include <stdio.h>

 void main () {

int V[20],i; int *p; p=V;

for(i=0;i<3;i++)

{

printf ("upisi %d. broj u vektor :\n",i);

scanf("%d",&V[i]);

}

for (i=0;i<3;i++)

{

*p=V[i];

p++;

}

p=V;

for (i=0;i<3;i++)

{

printf ("%d. broj u vektoru je:%d\n",i,*p);

p++;

}

printf ("%d brojeva smo ispisali pomoću pokazivaca",i);

}
```

40. Što je polje? Na primjeru pokažite deklariranje 1D i 2D polja.

Polje je skup od dvije ili više varijabli s istim imenom koje se označavaju brojem u zagradi (indeksom), a služe za pohranjivanje niza istovrsnih podataka.

Pr.: 1D: int x[20], y[15];

2D: int tablica [10] [5];

41. Pokažite na primjerima inicijaliziranje pri deklaraciji i u programu, te objasnite razliku.

Inicijaliziranje pri deklaraciji: int x[5]={3, 7, 5, 12,9};

Inicijaliziranje u programu: int x[n], i;

```
for(i=0;i<n;i++)

{

scanf("%d",&x[i]);

}
```

Razlika je u tome što kod inic. pri deklaraciji odmah određujemo vrijednosti koje se nalaze u polju, a kod inic. u programu možemo poslije unijeti vrijednosti koje hoćemo u to polje.

42. Na kratkom primjeru pokažite pretraživanje i sortiranje elemenata polja.

```
int sort;

do

{

sort = 0;

for (i=0; i<n-1;i++)                //za svaki element x[i] provjerava se da li je veći

if (x[i] > x[i+1])                  // od slijedećeg elementa

{                                   //ukoliko je veći njegova vrijednost se sprema u

float temp;                         // temp varijablu te mu se pridružuje vrijednost
```

```
temp = x[i]; // slijedeće varijable, dok se slijedećoj varijabli

x[i]=x[i+1]; // prodružuje vrijednost varijable temp odnosno

x[i+1]=temp; // prethodne varijable

sort = 1; //sort =1 sve dok se rade izmjene. Ukoliko nije //napravljena niti jedna izmjena sort
=0 i do-while petlja se završava

}

} while (sort ==1);
```

43. Što je struktura i kako se deklarira? Pokazati i na primjeru.

Struktura je skup od jednog ili više tipova varijabli, omogućava grupiranje podataka u jednu cjelinu.Strukture predstavljaju sredstvo za definiranje korisničkih tipova, a definiraju se pomoću struct ključne riječi:

```
struci kupac

{

char ime[20];

char adresa[30];

int id; float dug;

float narudzba;

}kupl, kup2;
```

44. Pokazati na primjeru što je polje strukture.

Polje strukture predstavlja sredstvo za čuvanje podataka koji se zapisuju ili učitavaju s diska. Pr.:

```
struct kupac

{

char ime[20];
```

```
char adresa[30];

int id;

float dug;

float narudzba;

} kup[500j;
```

45. Kako C rješava problem prenosivosti u smislu definiranja ulazno/izlaznih jedinica?

C ne posjeduje niti jednu ulazno-izlaznu naredbu, već su U-I operacije realizirane skupom funkcija iz standardne biblioteke čiji se funkcijski prototipovi nalaze u zaglavlju <stdio.h>. U-I funkcije su napisane posebno za svaki tip računala kako bi se osiguralo njihovo izvođenje na svim računalima, a C ne izvodi U-I operacije čitanjem ili zapisivanjem na stvarne sklopovske uređaje, već na C-ove standardne uređaje. O preusmjeravanju podataka na stvarne uređaje brine se OS.

46. Na primjerima pokazati korištenje formatirane printf funkcije.

```
#include <stdio.h>

main()

{

int x,y,z;

printf ("upiši brojeve x i y:\n");

scanf ("%d %d",&x,&y);

z=x+y;

printf ("zbroj brojeva %d i %d iznosi %d",x,y,z);

}
```

47. Na primjerima pokazati korištenje formatirane scanf funkcije.

*isto kao zadatak iznad

48. Prikazati otvaranje i zatvaranje datoteke u C programu.

```
#include <stdio.h>
 void main ()
{
int n,i;
char ime[20],pre[15];
F1LE *pFile;
pFile=fopen ("Dat.txt",V);
fscanf(pFile,"%d",&n);
fclose (pFile);
pFile=fopen ("Student.txt","w");
for (i=0;i<n;i+-+)
{
printf ("molimo unesi neko ime:\n");
scanf ("%s%s",&ime,&pre);
fprintf(pFile,"%s %s\n",ime,pre);
}
fclose (pFile);
}
```

49. Navedite dvije osnovne naredbe koje u C-u podržavaju ulaz i izlaz podataka.

U C-u ulaz i izlaz podržavaju u paru naredbe scanf (ulaz) i printf (izlaz).

50. Navesti i ukratko objasniti barem tri zaglavlja standardne biblioteke.

math.h => deklarira matematičke funkcije i makroe (sin, cos, sqr,...)

stdio.h => definira tipove, makroe i std. U-I tokove stdin, stdout, stderr i deklarira funkcije U-I toka (fopen,fprintf,fsetpos,...)

stdlib.h => deklarira nekoliko općih funkcija: za pretvorbu tipova, pretraživanje, sortiranje (atof, mailoc, abort, labs,...)

string.h => deklarira nekoliko funkcija za manipuliranje stringovima i memorijom (strcpv, strstr, memchr, memset,...)

locale.h => deklarira funkcije koje osiguravaju informacije zavisne o zemlji i jeziku