1. U switch strukturi:

- switch započinje grananje,
- case navodi konkretnu vrednost za koju se izvršava deo koda,
- break prekida izvršavanje grane,
- return izlazi iz funkcije.

---

2. Funkcija fgets je namenjena čitanju jednog reda teksta iz datoteke.

Ona prekida čitanje u tri slučaja:

- kada naiđe na znak za novi red \n (kraj reda),
- kada učita maksimalan broj karaktera (ovde 80),
- kada dođe do kraja datoteke.

---

3. fprintf zahteva:

- gde piše (datoteka)
- format
- podatak

stdout je standardni izlaz (ekran).

---

4. Strukture funkcionišu tako što:

- Struktura u strukturi znači da su potrebne dvostruke vitičaste zagrade.
- Prvo ide Tacka3D, zatim R.

Ostali odgovori imaju pogrešan redosled ili sintaksu.

---

5. fread ima oblik:

fread(adresa, veličina, broj_elemenata, datoteka);

3 bajta × 1 element = 3 bajta

Koristi se &podatak jer je potrebna adresa (fread radi sa memorijom zato uvek traži adresu podataka).

fscanf se ne koristi za binarne datoteke.

---

6. fwrite ima oblik:

fwrite(adresa, veličina, broj_elemenata, datoteka);

3 bajta × 1 element = 3 bajta

Koristi se &podatak jer je potrebna adresa (fwrite radi sa memorijom zato uvek traži adresu podataka).

fprintf je za tekst, ne za binarne fajlove.

---

7. Prikaz podataka funkcioniše na sledeći način:

- %f uvek ispisuje 6 decimala.
- %g ispisuje kraći zapis (bez nepotrebnih nula).

---

8. Prioritet operatora:

10 > 5 → tačno, dakle 1

1 == 1 → tačno, dakle 1

U C-u relacioni operatori imaju veći prioritet od ==, zato se izraz tumači kao:

x = (1 == (10 > 5))

što daje 1 == 1, što je tačno, odnosno 1.

---

9. Logički operator ILI:

Pošto je 10 != 5 tačno, sledi:

tačno || bilo šta - uvek daju tačno.

---

10. Najjasniji i najčitljiviji način:

Paran broj znači da je ostatak pri deljenju sa 2 jednak 0.

Ostali odgovori imaju pogrešnu logiku ili koriste bitovske operatore bez zagrada (zamka prioriteta).

---

11. Promenljiva y dobija vrednost 4 samo ako su oba uslova ispunjena:

- x > 1
- x < 6

Zato se uslovi povezuju sa AND (&&).

---

12. Petlja se izvršava dok:

- je j < n
- i dok su elementi pozitivni
- čim naiđe negativan ili nulu petlja se prekida (break)

To se tačno preslikava u uslov while(j<n && a[j]>0).

---

13. Petlja se završava kada b postane 0, a posle petlje važi: b == 0 .

Zatim:

- y = b; // y = 0
- x /= y; // deljenje nulom uvek dovodi do greške

14. Deo koda funkcioniše tako što:

- pod & 1 proverava poslednji bit
- >>= 1 pomera bitove udesno
- svaki put kada je bit 1, brojač br se uvećava

---

15. n & 1 je tačno za neparne brojeve (operator & (bitovsko I) proverava poslednji bit broja u binarnom zapisu, a svaki ceo broj u binarnom zapisu završava se na 0 ako je paran broj ili na 1 ako je neparan).

Dok je uneti broj neparan unos se ponavlja, a završava tek kad je unet paran broj.

---

16. Ovo je ciklično pomeranje ulevo za 1:

- prvi element sačuva u temp
- svi ostali se pomere jedno mesto ulevo
- poslednji postane stari prvi (temp)

---

17. Pošto se rezultat poredi sa NULL, funkcija mora da vraća pokazivač (ne int ili char).

Argumenti lista[i] i ime se prosleđuju kao nizovi karaktera što znači da su potrebni pokazivači na char (char[] / char*).

Od ponuđenih, samo 3. ima povratnu vrednost pokazivača i nizove za parametre.

---

18. Ako je mat float**, mat[i] će pokazivati na i-ti element (koji je tipa float*).

Funkcija mora vratiti float* (što znači da mora počinjati sa *Formiraj).

Parametri: x je int, a 0.5 float (što znači da u zagradama funkcije mora biti taj redosled).

---

19. Tok izvršavanja ide ovako:

- k = 4 - ispiše 4, pozove prikaz(5,10)
- k = 5 - ispiše 5, pozove prikaz(6,10)
- …
- k = 10 - ispiše 10, više ne ulazi u if (k < n je netačno)

Kada k = 10 završi:

- funkcija se vraća tamo gde je pozvana (k = 9)
- sada se izvršava donji printf
- ispisuje 9
- zatim se vraća na k = 8, pa ispisuje 8
- … sve do 4

---

20. Deklarisana funkcija traži:

- x traži int* (adresu promenljive), adresa od a je &a
- y traži int* (isto adresu promenljive), adresa od b je &b
- p traži int** (adresu pokazivača), adresa *poc je &poc

---

21. p->osnova.brojTemena = 6; je tačno jer:

- p je pokazivač, do polja strukture se ide sa ->
- osnova je „unutrašnja" struktura, a zatim .brojTemena

Pravilo je: pokazivač->polje (ne pokazivač.polje)

22.

1. netačno: while radi dok je uslov tačan (ne dok je netačan).

2. netačno: kad znaš tačan broj ponavljanja češće se koristi for.

3. tačno: kod while se prvo proverava uslov, pa se tek onda telo izvršava.

4. tačno: moguće je da se telo ne izvrši nijednom (ako uslov odmah nije ispunjen).

23.

1. tačno: koristi se kada se ne zna unapred koliko puta će se ponavljati.

2. netačno: ponavlja se dok je uslov tačan, ne dok je netačan.

3. netačno: ciklus se završava kada uslov postane netačan.

4. tačno: do-while se izvrši bar jednom (uslov je na kraju).

24.

3. ako datoteka ne postoji fopen vraća NULL

4. ako postoji fopen vraća pokazivač na datoteku.

25.

1. netačno: int a=b=c=5; - b i c nisu deklarisani u toj naredbi.

2. tačno: int a=5, b=5, c=5; - ispravan način deklarisanja većeg broja promenljivih.

3. netačno: char zn="a"; - "a" je string, a potreban je karakter 'a' (po jedan umesto dva navodnika).

4. netačno: long a; b=5; c; - sintaksno i logički neispravno (nakon ; potrebno je navesti ponovo tip podatka).

5. tačno: int a=0xf2; - heksadekadna vrednost se može koristiti.

6. tačno: char zn='\b'; - karakter (escape) se može postaviti za vrednost karaktera.

26.

x %= y; Operator %/%= važi samo za celobrojne tipove.

x =/ y + 5; =/ ne postoji kao operator (treba x /= (y + 5);).

y = x + z; y je const, ne može da se menja dodelom.

27.

scanf("%d%f", &a, &b); %f traži pokazivač na float, a dat je &b (int*).

scanf("%d%*d", &a, &b); %*d ne uzima argument, a ovde je dat višak &b.

scanf("%d%d", a, b); nedostaju & kod a i b (trebaju adrese).

---

28. Tačni:

int a[10]={1,2,3}; ostalo (do 10 broja članova) se popuni nulama.

int a[]={10,20,30,40,50}; dužina se sama odredi na osnovu broja članova (5).

int a[5]={'1','2','3','4','5'}; char konstante su samo brojevi (u ASCII kodu), mogu u int.

Netačni: 2 (previše inicijalizatora), 4 (pogrešna sintaksa), 5 (nije niz).

29.

Tačni: Dodela jednog pokazivača drugom, oduzimanje i upoređivanje dva pokazivača (u praksi „ispravno” kad su iz istog niza/bloka), upoređivanje pokazivača sa NULL.

Netačni: 2 (ne dodaješ realan broj na pokazivač), 4 (ime niza nije promenljiv pokazivač), 6 (ne sabiraju se dva pokazivača).

30.

Tačni: int a[k][k]; (k se može koristiti jer je fiksna dimenzija) int c[k][10]; int x[100][50];

Netačni: 5 (nije 2D niz) 2 i 6 koriste promenljivu m koja nije fiksna dimenzija.

31.

1. (9 / 5) * tempc + 32 - jer je 9/5 celobrojno deljenje pa je rezultat 1.

2. 9 / 5 * tempc + 32 - isto (9/5 je opet 1).

3. 9 * tempc / 5 + 32 - tempc je float, pa se računa realno (kao decimalna vrednost)

4. 32 + 9 * tempc / 5 - isto, samo drugačiji raspored.

32.

1. Ima default: p--;

2. Nema default, što sprečava da se p—izvrši za vrednosti koje nisu obuhvaćene case-ovima.

3. case 3: case 5: + default: p--; je tačno

4. Nema break, pa za 3/5 produži u default i uradi i p--.

33.

1. printf("%f", B[i]); - tačno jer je B[i] vrednost tipa float, a %f ispisuje realan broj.

2. printf("%f", &B[i]); netačno jer je &B[i] adresa (float*), a %f ne ispisuje adrese.

3. printf("%f", B+i); - netačno jer je B+i pokazivač (float*), a %f očekuje realnu vrednost.

4. printf("%p", *(B+i)); - netačno jer je *(B+i) realna vrednost (float), a %p služi za ispis adresa.

5. printf("%f", *(B+i)); tačno jer je *(B+i) isto što i B[i] – realna vrednost.

34.

1. scanf("%f", B[i]); - netačno jer je B[i] vrednost, a scanf zahteva adresu (float*).

2. scanf("%f", B+i);- tačno jer je B+i adresa i-tog elementa (float*).

3. scanf("%p", B+i); - netačno jer %p služi za unos adrese, a ne realnog broja.

4. scanf("%f", &B[i]); - tačno jer je &B[i] adresa i-tog elementa i odgovara formatu %f.

5. scanf("%f", *(B+i)); - netačno jer je *(B+i) vrednost tipa float, a ne adresa.

35.

Tačno: 1, 2, 4, 6

Netačno: 3 (loša sintaksa parametara), 5 (2D niz bez definisane dimenzije kolona), 7 (nedostaju tipovi za b i c), 8 (funkcija ne može vraćati niz (float[]), već eventualno float*)

36.

1. tačno: fopen(...,"w") kreira datoteku ako ne postoji i vraća FILE*.

2. netačno: za "w" se datoteka kreira; NULL se vraća samo ako otvaranje nije uspelo (npr. prava, putanja).

3. netačno: standardna C biblioteka ne daje automatsko upozorenje; samo otvara/trunkira.

4. netačno: ne „puca” sama od sebe — fopen vrati NULL samo ako nije uspelo; program puca samo ako ti pogrešno koristiš taj rezultat.

5. tačno: "w" truncira (postavlja dužinu na 0) bez upozorenja.

37.

1. tačno: niz se u izrazu ponaša kao pokazivač na prvi element (int*).

2. netačno: &niz nije pokazivač na element (int*), već pokazivač na ceo niz (int (*)[10]).

3. netačno: koristi se adresa celog niza umesto adrese prvog elementa.

4. tačno: niz predstavlja početnu adresu bafera, sizeof niz daje tačnu veličinu celog niza (10 * sizeof(int)) i čita se jedan blok odgovarajuće veličine u niz.

5. netačno: poslednji argument mora biti FILE * (*in je objekat tipa FILE, a ne pokazivač).

6. netačno: fread zahteva četiri argumenta a ovde ih je samo tri.

38.

1. x.ocene[i] – tačno jer ocene je niz u strukturi, indeksiranje je validno.

2. *x.razred - netačno, x.razred je int, ne pokazivač (ne može da se dereferencira).

3. x->ime - netačno, -> se koristi samo kad je levo pokazivač; x nije pokazivač.

4. x[i].ocene – netačno, x nije niz struktura pa x[i] nije validno.

5. x.ime – tačno jer je pristup članu strukture preko . ispravan.

39.

1. *p->kilometraza – netačno jer se izraz tumači kao *(p->kilometraza), a kilometraza je tipa int, ne pokazivač, pa se ne može dereferencirati.

2. (*p).kilometraza – tačno jer se najpre dereferencira pokazivač p, a zatim se operatorom . pravilno pristupa polju strukture.

3. &p->kilometraza – netačno jer se ovim izrazom ne pristupa polju, već se dobija adresa polja. U ovom zadatku traže se načini pristupa poljima strukture, a ne dobijanja njihove adrese.

4. p->start – tačno jer se operatorom -> pravilno pristupa polju tipa niz (char[50]), koje se u izrazu ponaša kao pokazivač na prvi element.

5. *(p).start – netačno jer operator . ne može da se primeni na pokazivač p. Ispravno bi bilo (*p).start ili p->start.

40.

1. netačno: Saberi(m, y[i], y[i+1]); - y[i] i y[i+1] su int, a treba int*.

2. tačno: Saberi(y[i], x[i], x[i+1]); - prvi argument je int a x[i] i x[i+1] su int*.

3. netačno: Saberi(m, y, x[i][j]); - x[i][j] je int, a treći parametar mora biti int*.

4. netačno: Saberi(y, x[i], x[i+1]); - y se pretvara u int*, a prvi parametar mora biti int.

5. tačno: Saberi(10, y, x[0]); - y → int*, x[0] (prvi red) → int*.

6. tačno: Saberi(x[i][j], x[i], x[j]); - tipski je ispravno: x[i][j] je int, x[i] i x[j] su int*.

41.

1. Umetni(s2, s1[i]); – tačno, jer je s2 tipa char*, a s1[i] je tipa char.

2. Umetni(s2, s1); – netačno, jer je drugi argument s1 tipa char*, a treba char.

3. Umetni(s2, 'A'); – tačno, jer je 'A' tipa char.

4. Umetni(s1, s3); – tačno, jer se s1 u pozivu tretira kao char*, a s3 je char.

5. Umetni(*s2, s3); – netačno, jer je *s2 tipa char, a prvi argument mora biti char*.

6. Umetni(s3, &s1); – netačno, jer je s3 tipa char (treba char*), a &s1 je pokazivač na niz (char (*)[20]), ne char.

42.

x = 123 – jer %3i učita najviše 3 cifre: "123".

y = 45 – jer drugi %3i nastavlja odmah posle toga i učita sledeće cifre "45" (staje na razmaku).

43.

a = 4 – jer ++a prvo uveća a sa 3 na 4.

b = 3 – jer posle toga b %= 4 znači b = 15 % 4 = 3.

44.

x = 10 – jer je uslov x>50 netačan, pa se x-=10 ne izvrši.

y = 30 – jer y+=10; nije u vitičastim zagradama, pa se izvršava uvek (uslov x>50 nije povezan sa tom linijom koda).

45.

1) izbor=3 - x = 109 jer case 3 doda 3, zatim produži do prvog break-a kroz default koji postavi 100, pa case 4 doda 4 i case 5 doda 5.

2) izbor=10 - x = 109 jer ide direktno u default (100), pa produžuje do break-a i dodaje 4 i 5.

3) izbor=4 - x = 9 jer case 4 doda 4 pa zbog izostanka break-a produži u case 5 (koji dodaje 5).

4) izbor=2 - x = 2 jer case 2 dodaje 2 i break prekida.

46.

Poslednja kolona ima indeks M-1. Menjamo element u toj koloni za svaku vrstu x (od 0 do N-1), pa svaki mat[x][M-1] množimo sa 2.

---

47. Da bi se obrisao element p[k], svi elementi desno od njega se pomere jedno mesto ulevo (prepisuju se preko njega). Posle pomeranja, logička dužina niza se smanji (n--).

---

48. Prvi red ima indeks 0. Prolazimo kroz sve kolone k = 0..M-1 i dupliramo elemente mat[0][k].

49.

1. 10 / 4. = 2.5 - 4. je realan broj → računa se realno deljenje.

2. 10. / 5 = 2.0 - 10. je realan broj → rezultat je realan.

3. -10 % 3 = -1 - U C-u ostatak ima znak levog operanda (-10), pa je -10 = (-3)*3 + (-1).

4. 10. % 5 = greška - Operator % važi samo za celobrojne tipove, ne za realne (10. je double).

5. 10 % (-3) = 1 - 10 = (-3)*(-3) + 1, ostatak prati znak levog operanda (10 je pozitivan).

6. (100/3) % 6 = 3 - 100/3 je celobrojno deljenje → 33, a 33 % 6 = 3.

---

50. Sačuva se poslednji element (p[n-1]) u pom, zatim se svi ostali elementi pomere jedno mesto udesno (kreće se od kraja da se ne pregazi podatak), a na početak niza (p[0]) se upiše sačuvani element.

---

51. Kreće se od kraja (i=n) i pomera se udesno (p[i]=p[i-1]) dok se ne oslobodi mesto na k, pa se x ubaci u p[k].

---

52. Niz razlika je: +1, +2, +3, +4, +5… pa važi: A[i]=A[i−1]+i

---

53.

p2 pokazuje na x (jer je p2=p1, a p1=&x)

*p2 je vrednost x, tj. 40

54.

int a[7]={10, 25, 30, 15, 40, 77, 45}, *pa, x, y;

pa = a + 4; - pa -> a[4] (40)

x = --(*pa) + 5; - a[4] postaje 39, x = 44

y = *(--pa) + 5; - pa -> a[3] (15), y = 20

55.

int a[7]={81, 12, 35, 97, 40, 52, 17}, *pa, x, y;

pa = a + 3; - pa -> a[3] (97)

x = *(pa-2) + 1; - pa-2 -> a[1] (12), x = 13

y = (*pa - 2) + 1; - (97-2)+1 = 96

---

56. z je kopija (c se ne menja). U funkciji z++ napravi z=11, pa *x = *x + z menja a na 21, a (*y)++ uveća b na 11.

57.

Početno:

s1 = "Short Message Service"

s2 = strchr(s1,'M') → pokazuje na "Message Service"

s3 = strrchr(s2,'S') → poslednje S u tom delu → "Service"

strncpy(s1+1, s2, 1) → u s1[1] upiše 'M'

strcpy(s1+2, s3) → od s1[2] upiše "Service\0"

Konačno:

s1 = "SMService"

s2 = "ice" (jer i dalje pokazuje na staru poziciju gde je sada slovo i)

s3 = "Service" (ostao da pokazuje na staro “Service” u nizu)

58.

levo poravnanje → - (3.)

ispis znaka plus za pozitivan broj - + (4.)

dopuna nulama umesto razmacima - 0 (2.)

prisilna decimalna tačka i kad nema razlomka - # (1.)

59.

printf("%g", w); - ispis bez suvišnih nula 3. (123.456)

printf("%f", w); - ispis sa 6 decimala 1. (123.456000)

printf("%.2f", w); - ispis ograničen na dve decimale 4. (123.46)

printf("%e", w); - ispis koji koristi simbol e 2. (1.234560e+002)

---

60. sqrt predstavlja koren a fabs apsolutnu zagradu.

---

61. Program će ispisivati sve od početnog ispisa koji odgovara unetom znaku dok god ne dođe do break-a.

62.

int *a; - 3. (pokazivač na int)

int a[100]; - 2. (vektor/niz int)

int a*[100]; - 5. (sintaksna greška * ne može ići između naziva i uglastih zagrada)

int *a[100]; - 1. (niz pokazivača na int)

63.

&a[0] – 3. (adresa početnog elementa niza)

*(a+n-1) – 1. (vrednost poslednjeg elementa)

a+4 – 6. (adresa petog elementa, jer je to pomeraj za 4 int-a)

*a – 5. (vrednost prvog elementa)

64.

mat[j][n-1] – 1. (element u j-toj vrsti i poslednjoj koloni)

mat[j] - 3. (j-ta vrsta matrice)

mat[0][j] – 5. (element u prvoj vrsti i j-toj koloni)

mat[i] – 2. (i-ta vrsta matrice)

65.

ftell(dat) – 3. (vraća poziciju u bajtovima od početka)

fseek(dat, 0, SEEK_END) – 2. (pozicionira na kraj)

fseek(dat, 0, SEEK_SET) – 1. (pozicionira na početak)

rewind(dat) – 1. (takođe pozicionira na početak)

66.

short – 7. (%hd)

signed int (u dekadnom obliku) – 1. (%d)

long – 4. (%ld)

unsigned – 8. (%u)

signed int (dekadni, heksadekadni ili oktalni oblik) – 2. (%i)

67.

učitavanje karaktera iz datoteke – 5. (fgetc)

učitavanje reda iz datoteke – 2. (fgets)

formatirani upis podataka u datoteku – 4. (fprintf)

upis stringa u datoteku – 3. (fputs)

formatirano učitavanje podataka iz datoteke – 1. (fscanf)

68.

vraćanje na početak reda (carriage return) – 3. (\r)

sistemski zvučnik (bell) – 6. (\a)

prelaz u novi red (new line) – 1. (\n)

nije escape sekvenca – 5. (\h)

horizontalni tabulator – 2. (\t)

vraćanje jednu kursorsku poziciju nazad (backspace) – 4. (\b)

69.

Da li je c štampajući znak (uključujući i razmak)? – 6. (isprint(c))

Da li je c veliko slovo? – 4. (isupper(c))

Da li je c znak interpunkcije? - (nije ponuđena sekvenca za znak interpunkcije)

Da li je c upravljački znak? – 5. (iscntrl(c))

Da li je c decimalna cifra? – 2. (isdigit(c))

Da li je c beli znak? – 1. (isspace(c))

Da li je c slovo? – 3. (isalpha(c))

Da li je c heksa-dekadna cifra? - (nije ponuđena sekvenca za heksadecimalnu šifru)

70.

sn = strrchr(s1,'a') - 1;

Poslednje 'a' je u "again", jedno mesto pre je 'g' tako da je rezultat: "gain"

sn = strchr(s1,'a') + 1;

Prvo 'a' je u "again", pomeraj za +1 daje "gain" tako da je rezultat: "gain"

sn = strstr(s1,"my");

Pronalazi "myM8sagain"

sn = strstr(s1,"T2");

"T2" ne postoji u stringu tako da je rezultat: NULL

---

71. Promenljive definisane unutar metoda važe samo u okviru tog metoda i ne mogu se koristiti van njega. U C# terminologiji to su lokalne promenljive.

---

72. Članovi klase u C# mogu pripadati:

- klasi (static) – zajednički za sve objekte
- objektu (nestatički / instancni) – svaki objekat ima svoju kopiju

---

73. Statička polja postoje u jednoj kopiji na nivou klase i dele ih svi objekti te klase. Promena vrednosti važi za sve instance – svi gradovi jedne države imaju zajedničko polje država.

---

74. Instancna (nestatička) polja:

- vezana su za konkretan objekat
- svaki objekat ima svoju vrednost
- ne mogu se koristiti bez kreiranja objekta

Ostale tvrdnje opisuju statička polja.

---

75. U C#:

- pri kreiranju objekta izvedene klase,
- uvek se prvo poziva konstruktor roditeljske klase,
- zatim konstruktor izvedene klase (čak i ako base nije eksplicitno naveden — tada se poziva podrazumevani konstruktor roditelja).

---

76. Privatni (private) članovi roditeljske klase nisu dostupni izvedenoj klasi, pa im se ne može pristupiti ni preko base.

---

77. Metode se ne mogu razlikovati samo po povratnom tipu.

Ove dve verzije fun(int n) imaju isti potpis (ime + parametri), što izaziva grešku pri kompajliranju jer program ne može odrediti koju verziju metode da pozove.

---

78. U C# jeziku, opseg važenja (scope) promenljive i je samo telo te for petlje.

To znači:

- i postoji samo dok se petlja izvršava
- čim se petlja završi, i više ne postoji

---

79. BinarySearch ne sortira niz automatski.

Ako niz nije sortiran u rastućem poretku, rezultat je nepredvidiv (ne garantuje tačan indeks).

80.

this.x = x; this.y = y; - dodeljuje vrednosti poljima (this znači pripadanje klasi/objektu)

set(x, y); - poziva metodu koja to isto radi

Ostale opcije ne menjaju vrednosti polja objekta.

---

81. Metod nema bazni slučaj (npr. if(n==1) return 1;).

Zato se pozivi nastavljaju unedogled (fun(3)->fun(2)->fun(1)->fun(0)->...) i dolazi do greške (stack overflow / beskonačan lanac poziva).

---

82. U fun(2):

- n se nikad ne menja u while-petlji.
- Za n=2 stalno ispisuje (2-1) tj. 1, pa poziva fun(1) (koji odmah stane), i opet ponavlja.

---

83. Palindrom mora imati isti prvi i poslednji karakter.

Ostale opcije ili porede pogrešne indekse (s.Length je van opsega) ili pogrešan karakter (s[1]).

---

84. Kada su krajnji karakteri jednaki, sužavaš problem tako što pomeraš levu granicu udesno i desnu ulevo.

---

85. Ako je traženi broj veći od srednjeg, tražiš u desnoj polovini, pa novi levi postaje sredina + 1, a desni ostaje isti.

---

86. U Main se radi Test t = null; i zatim Console.WriteLine(t.x);.

Pošto je t null, pristup instancnom polju x baca NullReferenceException (greška u izvršavanju).

---

87. getBroj() vraća polje broj (instancno/nestatičko polje), pa ne sme biti static.

kvadrant(int n) koristi samo parametar n, ne koristi stanje objekta, pa može i ne mora biti static.

---

88. U copy konstruktoru treba kopirati koordinate iz objekta p.

Metoda Set(double,double) već to radi.

Pošto smo u istoj klasi, imamo pravo da pristupimo privatnim poljima p.x i p.y.

Poziv Set(p) ne odgovara broju parametara, a ključna reč this se ne može koristiti sama po sebi kao u odgovorima 1 i 2.

---

89. Poziva se a1.Equals(a2) gde su a1 i a2 tipa KlasaA, pa se koristi metoda Equals(KlasaA a).

Polje x nije inicijalizovano eksplicitno pa ima podrazumevanu vrednost 0 u oba objekta, zato je 0 == 0 odnosno true.

---

90. a1 i a2 su deklarisani kao Object, pa se poziva Object.Equals(object).

Klasa KlasaA nije override-ovala Equals(object), pa ostaje podrazumevano poređenje referenci odnosno dva različita objekta daju false.

---

91. Program nema greške; u klasi B nije override metoda Metod(int), nego je samo dodat preopterećen metod (metod sa drugačijim tipom parametra) Metod(string), pa se b.Metod(5) normalno poziva iz klase A.

CitajI() vraća i koje je postavljeno na 5.

---

92. catch (Exception e) je preširok i hvata sve greške, pa je sledeći catch (ArithmeticException ae) nedostižan (unreachable) što znači da kompajler prijavljuje grešku zbog pogrešnog redosleda catch blokova.

---

93. Ako ima više catch blokova i među njima i osnovni catch blok (Exception), on mora biti poslednji, jer je najopštiji (hvata sve greške pa nema smisla da se hvatanje specifičnih grešaka nalazi ispod).

94.

1. tačno: try mora imati bar jedan catch ili finally

2 tačno: try može imati više catch blokova

3. netačno: Exception ne ide u prvi, nego u poslednji (ako postoji više catch-ova)

4. tačno: redosled catch blokova je bitan (specifičniji pre opštijeg)

5. netačno: finally nije obavezan

6. netačno: može imati više catch blokova

95.

virtual omogućava da se metoda override-uje u izvedenoj klasi.

override je samo prepisivanje u izvedenoj klasi.

abstract zahteva da naslednici implementiraju (override-uju) metodu.

new samo sakriva, sealed sprečava dalje override, base nije modifikator metode, a public/protected sami po sebi ne omogućavaju redefinisanje.

96.

1. netačno: int niz = new int(30); - int nije niz, a i new int(30) je pogrešna sintaksa.

2. tačno: double[] niz = new double[30]; - tip + new + veličina u [].

3. tačno: int[] niz = { 3, 4, 3, 2 }; - ispravan način inicijalizacije niza.

4. netačno: new char[]; - ne može bez veličine u uglastim zagradama ili članova niza nakon njih.

5. netačno: new char { ... } - fali [] (mora new char[] { ... }).

6. tačno: char[] niz = new char[] { 'a', 'b' }; - eksplicitno new char[] + nabrajanje članova.

---

97. Gledaju se samo klase Sin i Deda jer klasa Sin nasleđuje klasu Deda, a od polja samo public i protected.

---

98. Gledaju se klase Sin, Otac i Deda jer klasa Sin nasleđuje klasu Otac koja nasleđuje klasu Deda, a od polja samo public jer se u pitanju radi o objektu klase.

99.

1. tačno: this.x = x; this.y = y; - dodeljuje vrednost poljima objekta.

2. netačno: x=x; y=y; — dodela parametara samima sebi, polja se ne menjaju.

3. tačno: set(x, y); - poziva metodu koja dodeljuje vrednost poljima objekta.

4. netačno: set(this.x, this.y); — prosleđuje postojeće vrednosti polja (0,0), ne parametre.

5. netačno: x=this.x; y=this.y; — menja parametre, ne polja.

100.

1. netačno: koristi b, sum3, i b[i].Length (to je za niz nizova int[][], ne za int[,]).

2. tačno: foreach radi direktno preko svih elemenata 2D matrice (int[,]).

3. tačno: ispravno obilazi vrste i kolone preko GetLength.

4. netačno: koristi b i foreach (int[] vrsta in b) (niz nizova), ne int[,].

---

101. Važno je da se params nalazi poslednji u listi paremetara (u zagradama), odnosno da nema ničega nakon params-a.

102.

1. tačno: this.Fun() može kao Fun() (nema konflikta imena, poziva se isti metod instance).

2. netačno: this.x ne može kao x jer bi x bio parametar konstruktora; to nije isto.

3. tačno: Poziv drugog konstruktora this(23) mora biti prva naredba u konstruktoru (u C# se piše u zaglavlju: public TestPrimer() : this(23) { ... }).

4. netačno: this(23) je ispravno jer se int 23 može implicitno konvertovati u double (postojeći konstruktor prima double).

103.

1. netačno: Podrazumevani konstruktor se ne dodaje uvek (ne dodaje se ako postoji bilo koji konstruktor).

2. tačno: Dodaje se samo ako nijedan konstruktor nije definisan.

3. netačno: Ne moraš eksplicitno definisati konstruktor.

4. tačno: Konstruktor nema povratni tip, ni void.

104.

4. netačno: privatni članovi nisu dostupni izvedenoj klasi.

5. tačno: konstruktor klase, konstruktori se ne nasleđuju (pozivaju se preko base(...)).

105.

1. netačno: Console.WriteLine(a) već interno poziva ToString(), ne moraš ručno.

2. netačno: za a se ne poziva Object.ToString(), nego override u Klasa.

3. tačno: a je tipa Object, ali pokazuje na objekat Klasa, a ToString() je override-ovan pa se poziva Klasa.ToString().

4. tačno: obj je običan new Object(), pa se poziva Object.ToString().

106.

2. tačno: KlasaB ima podrazumevani konstruktor, ali on pokušava da pozove base(); pošto KlasaA nema konstruktor bez parametara, to je greška pri kompajliranju.

3. tačno: Greška se uklanja ako se u KlasaA doda konstruktor bez parametara KlasaA() (alternativno: u KlasaB eksplicitno pozvati base("neki tekst")).

---

107. Potrebno je obratiti pažnju na redosled parametara (dva string-a pa dva int-a) kao i na postojanje konstruktora bez parametara.

108.

U slučaju greške - catch

Izvršava se i kad ima i kad nema greške - finally

Naredbe koje mogu izazvati grešku - try

---

109. Potrebno je obratiti pažnju na to koja klasa je izvedena od koje.

base.Uvecaj() se odnosi na vrednost koju ta metoda vraća u baznoj klasi.

---

110. Potrebno je obratiti pažnju na to koja klasa je izvedena od koje.

base.Uvecaj() se odnosi na vrednost koju ta metoda vraća u baznoj klasi.

---

111. Metodu sa ključnom rečju *new* gledamo samo kada se radi o objektu klase u kojoj se metoda nalazi (i sa leve i sa desne strane pri kreiranju objekta je KlasaC), u svakom drugom slučaju gledamo metodu koja je *override* ukoliko takva postoji (ili u početnom slučaju metodu *virtual* jer je KlasaA i sa leve i sa desne strane pri kreiranju objekta).

---

112. Metodu sa ključnom rečju new gledamo samo kada se radi o objektu klase u kojoj se metoda nalazi u suprotnom gledamo da li postoji metoda koja je *override* i gledamo nju.

113.

Klasa koja se prostire u više fajlova – 3. (partial)

Klasa sadrži samo deklaracije metoda, ali ne i definiciju (telo) metode – 4. (interface)

Klasa koja se ne može instancirati – 1. (abstract)

Klasa iz koje se ne može nasleđivati – 2. (sealed)

114.

vidljiv unutar klase u kojoj je definisan, kao i unutar izvedenih klasa – 3. (protected)

vidljiv samo unutar klase u kojoj je definisan – 1. (private)

vidljiv unutar projekta u kome je definisan – 4. (internal)

vidljiv i van svoje klase u kojoj je definisan – 2. (public)

115.

Opis funkcionalnosti objekata klase – 4. (metod)

Kontrola pristupa poljima klase – 5. (svojstvo/property)

Opis osobina objekata klase – 1. (polje/atribut)

Kreiranje objekata klase – 3. (konstruktor)

Uništavanje objekata klase – 2. (destruktor)

---

174. Ukoliko se više elemenata sa z-indexom preklapa videće se onaj koji ima veću vrednost z-indexa.

199.

1. tačno: WSDL opisuje kako klijenti komuniciraju sa servisom (operacije, poruke, endpoint).
2. netačno: Način implementacije metoda nije deo WSDL-a (to je interna stvar servisa).
3. tačno: WSDL navodi koje operacije (metode) servis nudi, njihove ulazne i izlazne poruke.
4. netačno: WSDL ne definiše protokol (npr. HTTP, SOAP), već može da se veže za protokol.

202.

1. tačno: XSD definiše da li je nešto string, int, date, itd.

2. netačno: XSD ne propisuje konkretne vrednosti (osim ograničenja tipa).

3. tačno: Definiše koje elemente neki element sme da sadrži.

4. tačno: Može da zahteva redosled (sequence, choice, all).

5. netačno: Redosled atributa nije bitan u XML-u.

6. netačno: XML dokument ima samo jedan root element.

242.

1. netačno: Foreign key – služi za povezivanje tabele sa drugom tabelom i obezbeđivanje referencijalnog integriteta.

2. netačno: Unique – služi za obezbeđivanje da sve vrednosti u koloni budu jedinstvene.

3. tačno: Distinct – služi za uklanjanje duplikata u rezultatu SELECT upita.

4. netačno: Check – služi za proveru da li vrednosti u koloni ispunjavaju zadati uslov.

5. tačno: Convert – služi za konverziju jednog tipa podataka u drugi.

6. tačno: Union – služi za spajanje rezultata više SELECT upita u jedan skup rezultata.

7. netačno: Not Null – služi za zabranu unosa NULL vrednosti u kolonu.

8. tačno: Except – služi za izdvajanje redova iz prvog upita koji ne postoje u drugom upitu.