1) Који оператори се не могу преклапати, а који се могу преклопити само као методе класе?

Ne mogu se preklapati operatori :: , ?: , . i .* i operatori sizeof i throw.
Ne mogu biti static i ne mogu imati podrazumevane vrednosti.
Samo kao metode (ne i prijateljske funkcije) se preklapaju operatori =, ->, (), [] i (tip).
Izuzeci su operatori new i delete.

2) Да ли се заштићеном члану основне класе O може приступити из методе приватно изведене класе I и да ли му се може приступити из методе класе II која је јавно изведена из I? Образложити.

Iz I može, jer je pristup zaštićenim članovima dozvoljen iz metoda izvedene klase.
Iz II ne može jer privatnim izvođenjem član postaje privatan i metoda više nema pravo pristupa.

3) При вишеструком извођењу, којим редоследом се позивају конструктори основних класа?

Pozivaju se redom kojim su navedeni pri izvođenju, ne onim redom kojim su navedeni u inicijalizatoru.

4) Како треба писати преклопљени оператор доделе вредности за неку класу?

Piše se kada su atributi klase pokazivači
Operator = treba preklopiti kao nestatičku metodu te klase i to tako što se prvo obriše (uništi) stari sadržaj, a zatim se stvori novi kao kopija celog objekta sa desne strane operatora.

5) Каква је семантичка разлика између јавног и приватног извођења?

Pri javnom izvođenju: javni, zaštićeni i privatni članovi postaju javni, zaštićeni i privatni.
Pri zaštićenom izvođenju javni, zaštićeni i privatni članovi postaju zaštićeni, zaštićeni i privatni.
Pri privatnom izvođenju javni, zaštićeni i privatni članovi postaju privatni.
Javno izvođenje je slično relaciji jeste, a privatno i zaštićeno izvođenje je slično operaciji sadrži.

6) У односу на декларацију у основној класи, како мора изгледати декларација редефинисане методе у изведеној класи?

Deklaracija mora biti identična. Povratna vrednost*, ime i broj i tipovi argumenata moraju biti identični.
*Ako je povratna vrednost bio pokazivač na osnovnu klasnu, može da bude pokazivač na izvedenu.

7) Које особине стандардних оператора се подразумевају при њиховом преклапању?

Podrazumeva se asocijativnost (način grupisanja), prioritet i n-arnost.

8) Навести два начина кориснички дефинисане конверзије и основне разлике између њих?

Preko konstruktora (konvertuje se iz bilo kog tipa u tip klase kojoj pripada) i preklapanjem operatora (tip) (konverzija iz tipa kom pripada u bilo koji drugi standardni ili klasni tip).

9) Ако основна класа садржи један низ објеката, а изведена други низ – навести презизан редослед конструкције објекта изведене класе.

1. Konstruišu se objekti niza osnovne klase
2. Konstruiše se osnovna klasa
3. Konstruišu se objekti niza izvedene klase
4. Konstruiše se izvedena klasa

10) Који проблеми се појављују при вишеструком извођењу и како се решавају?

Dijamant struktura, odnosno izvedena klasa sadrži više puta inicijalizovanu osnovnu klasu u sebi. (nacrtati primer i detaljno objasniti)
Rešava se dodavanjem ključne reči **virtual** pri izvođenju klase, ispred svakog navođenja osnovne klase.

11) Да ли је исправан део програма: X x, *px, ℞ px=new X; rx=*px; и зашто?

Ne može da se napravi rx – mora odmah da se inicijalizuje.

12) Којим редоследом се конструишу, односно уништавају, атрибути класних типова? Да ли се конструкција, односно деструкција, атрибута класних типова обавља пре извршења тела конструктора, односно деструктора?

Konstruišu se onim redom kojim su navedeni u definiciji klase.
Konstrukcija se vrši pre izvršenja tela konstruktora.
Destrukcija se vrši nakon izvršenja destruktora.
Sva uništavanja idu obrnutim redosledom.

13) Да ли пријатељска метода има показивач this на објекат класе чији је пријатељ и зашто?

Nema, jer ona nije prijatelj jednom objektu nego celoj klasi.

14) Како и где се стварају (креирају) динамички објекти? Како се уништавају?

Na heap-u. Pomoću operatora delete.

15) Навести дефиницију типа показивача this у телу константне методе неке класе X.

Const X* this

16) Под којим условима конструктор копије може да има више формалних аргумената?

Ukoliko svaka od njih ima svoje podrazumevane vrednosti.

17) Да ли се статичка (заједничка) метода може позвати пре него што се направи први објекат одговарајуће класе и зашто?

Može jer ne sadrži this i jer nije vezan za konkretan objekat (nego za celu klasu).

18) Које особине има релација пријатељства између класа?

Nema this, nije simtrična, nije tranzitivna, ne nasleđuje se, menja pravo pristupa, ali ne i oblast važenja promenljive (može da joj se pristupa iz druge klase, ali joj ne pripada).

19) Шта је то *л*-вредност (*lvalue*)?

L-vrednost je vrednost koja ima svoju adresu, postoji u memoriji, i može da se piše sa leve strane operatora dodele (=).

20) Шта је улога, шта је операнд, а шта резултат операције за операторе new и delete?

Uloga je alokacija i dealokacija memorije.
Operand za new je identifikator tipa (T) i eventualni inicijalizatori.
Operand za delete je pokazivač na neki tip.
Povratna vrednost za new je pokazivač na zadati tip.
Delete nema povratnu vrednost (void).

21) Која је намена конструктора копије у класи X, којег типа је његов формални аргумент, а ког стварни?

Namena je da inicijalizuje objekat objektom iste klase (napravi njegovu identičnu kopiju)
Formalni argument je X&, a stvarni argument je X ili X&.

22) Да ли су дозвољени показивачи на референце? Зашто?

Nisu, jer reference nisu L-vrednosti.

23) За класу X написати декларације подразумеваног конструктора, конструктора копије и конструктора за конверзију из класе Y.

X () ;
X (const X&) ;
X (Y&) ;

24) Да ли се из локалне класе може директним именовањем приступити: (1) статичком локалном податку и (2) аутоматском локалном податку обухватајуће функције? Образложити.

Unutar lokalne klase iz okružujućeg dosega je dozvoljeno korišćenje:
- identifikatora tipova
- nabrojanih konstanti
- **statičkih lokalnih i globalnih objekata**
- spoljašnjih (eksternih) promenljivih i funkcija

1. Može.
2. Ne može.

25) Да ли апстрактна класа може имати конструктор и зашто?

Može, jer izvedene klase pozivaju njen konstruktor radi inicijalizacije eventualnih skrivenih atributa date osnovne apstraktne klase.

26) Да ли је исправно бацити као изузетак показивач на локалну променљиву текуће методе и зашто?

Ne, zato što lokalna promenljiva prestaje da postoji nakon izvršavanja metode.

27) Да ли је механизам генерика статички или динамички и зашто?

Mehanizam generika je statički jer je kreira u toku prevođenja.

28) Шта се дешава када се у некој catch грани изврши наредба throw; ?

Ukoliko su try blokovi ugnježdeni, izuzetak se prosleđuje hendleru sledećeg (višeg) nivoa naredbe try.

29) Која врста повезивања се примењује на имена анонимног простора имена и како се користе имена из анонимног простора?

Promenjuje se unutrašnje povezivanje i koriste se bez operatora za razrešenj dosega.

30) Шта се назива делимичном, а шта потпуном специјализацијом шаблона? Навести декларације као примере обе специјализације шаблона template <class T1, class T2> class S;

Specijalizacija je difinisanje ponašanja određenog šablona za konkrektne vrednosti parametara template-a.
Potpuna specijalizacija je ona specijalizacija kod koje su svi parametri šablona konkretizovani.
template <> class S;
Delimična specijalizacija je ona specijalizacija kod koje su samo neki parametri šablona konkretizovani.
template <class T1> class S;

31) Шта се дешава са изузетком баченим у неком try блоку уколико иза блока не постоји одговарајућа catch грана која може да обради изузетак?

Takav izuzetak se naziva neobrađeni iuzetak i on poziva funkciju terminate() koja poziva funkciju abort().
Postoji funkcija set_terminate() u kojoj je moguće preinačiti ponašanje funkcije terminate, tako što joj se predaje pokazivač na funkciju koja treba da se izvršava umesto abort().

32) Како се може спречити аутоматско генерисање функције из шаблона за неки тип аргумента?

Preklapanjem imena funkcije i definisanjem željenog ponašanja.

33) Шта представља показивач this, којег је типа и да ли се може користити у статичкој функцији чланици неке класе?

Predstavlja pokazivač na trenutni objekat. Tipa je const X*, gde je X posmatrana klasa i ne može se koristiti u statičkim funkcijama.

34) Навести редослед активности при конструкцији и деструкцији објеката (основних класа).

Videti prethodna pitanja.

35) У чему су разлике, а у чему сличности референце (упућивача) и показивача у језику C++?

Oba predstavljaju objekat, s tim što pokazivač zauzima mesto u memoriji na kom se čuva adresa objekta, a referenca je drugo ime za dati objekat. Pokazivač je L-vrednost, referenca nije. Referenca mora da se inicijalizuje pri definisanju i nije joj moguće kasnije promeniti vrednost. Ne mogu se napraviti nizovi referenci.

36) Да ли нека метода позвана за један објекат неке класе има право приступа приватној секцији другог објекта исте класе и зашто?

Ima, jer je metoda vezana za klasu a ne za konkretan objekat.

37) Колико конструктора и колико деструктора може имати нека класа и зашто?

Ne postoji ograničavajuči broj konstruktora, ima ih koliko je korisniku potrebno jer klasa može da se konstruiše na razne načine. Destruktor postoji samo jedan, nema povratne vrednosti i ne prima nikakve argumente.

38) У чему је основна разлика између иницијализације и доделе вредности? Како се ове две ствари прилагођавају конкретним потребама неке класе?

Inicijalizacija poziva konstruktor, a dodela vrednosti operator =.
Tako što korisnik napiše potrebne konstruktore i preklopi operator = za datu klasu.
*Takođe je moguće inicijalizovati objekat operatorom jednako
X x = 5;*

39) Да ли се у језику C++ могу редефинисати значења оператора за стандардне типове и зашто?

Ne, jer se jezik C++ može proširivati, ali ne i menjati.

40) Шта су референце (упућивачи) и да ли је дозвољено имати низ референци? Зашто?

Ne, jer da bi mogao da se definiše niz, mora da se definiše ipokazivač na njega, a to nije moguće. Reference su alternativna imena za neki podatak i ne zauzimaju mesto i memoriji. Pišu se sa znakom & i moraju da se inicijalizuju pri kreiranju jer nisu L-vrednosti.

41) Која општа ограничења постоје при преклапању оператора?

Ne mogu da se menjaju ponašanja operatora za standardne tipove. Nije moguće menjati n-arnost, asocijativnost i prioritet operatora.

42) Шта исписује приложени програм?

Ispisuje se:
~B: a[0]=0 a[1]=1 a1=2
~A:2
~A:1
~A:0

43) У којим ситуацијама има смисла да функција буде непосредно уграђена у код и на које начине се то може постићи код метода класе?

Kada je funkcija veoma kratka i njeno pozivanje traje duže od njenog izvršavanja. Postiže se pisanjem ključne reči inline pre definicije metode. Ukoliko se definicija funkcije nalazi u okviru definicije klase podrazumeva se inline.

44) Да ли се у методи класе X, која је пријатељска метода класе Y, може директним именовањем приступати заштићеним и приватним члановима класе Y? Зашто?

Ne, potrebno je napisati operator za razrešenje dosega, jer relacija prijateljstva između klasa ne utiče na doseg promenljivih već samo na pravo pristupa.

45) Како се преклапају операторске функције аутоинкрементирања и аутодекрементирања? Навести декларације одговарајућих функција.

Za preklapanje prefiksnih oblika operatora ++ i --
Kao metoda bez argumenata (T operator@@()), ili kao prijateljska funkcija sa argumentom tipa klase (T operator@@(T)).
Za preklapanje postfiksnih oblika operatora ++ i --
Kao metoda sa jednim argumentom tipa int (T operator@@(int)), ili kao prijateljska funkcija sa dva argumenta tipa int i klase (T operator@@(T, int)).

46) Да ли виртуелна метода може да промени throw() листу методе из базне класе и како?

Može da je suzi, ne i da je proširi.

47) Да ли конструктор може бити виртуелна функција и зашто?

Ne može, jer pri kreiranju mora eksplicitno da se kaže kog je tipa objekat koji treba napraviti.

48) Објаснити зашто наведена конструкција није исправна:

class A { int i; public: A(int ii){i=ii;} };
class B: public A { int j; public: B(int jj){j=jj;} };
Pošto klasa A nema podrazumevani konstruktor, konstruktor za klasu B nije kompletan.

49) Да ли апстрактна класа може имати атрибуте и конкретне методе и зашто?

Može, jer činjenica da je ona apstraktna znači samo da je bar jedna od metoda čisto virtualna, a ne da je cela čisto virtualna. Izvedena klasa može bez problema da nasledi regularne metode i atribute.

50) Да ли се у свим функцијама чланицама може користити показивач this? Образложити.

Ne može u prijateljskim i ne može u statičkim.

51) Каква је разлика између преклапања (overloading) имена и редефинисања (overriding) метода? Навести пример.

Kod preklapanja imena ne mogu da budu iste povratne vrednosti, broj argumenata i kog su tipa, već samo da se metode zovu isto.
Kod redefinisanja je bitno da budu isti broj argumenata ,tip, kao i ime metode, a povratna vrednost može da bude pokazivač na izvedenu klasu.

52) Навести тачан редослед активности при стварању објекта неке изведене класе D из основне класе B, где D садржи члан класе Y, а B садржи члан класе X.

X pa B pa Y pa D.

53) Ако је основна класа изузетака B, из ње изведена класа D1, а из D1 изведена класа D2, написати наредбу try са одвојеном обрадом (catch гранама) сва три типа изузетака (са празним телима рутина за обраду изузетака).

try{}
Catch (D2) {}
Catch (D1) {}
Catch (B) {}

54) Да ли уграђену (inline) функцију треба писати у .h или у .cpp фајлу и зашто?

Treba je pisati u .h fajlu zbog uštede u programerovoj energiji.

55) Дискутовати могућност приступа заједничкој (*static*) методи класе преко објекта те класе?

Moguće joj je pristupiti iz svakog objekta te klase, iako ona nastaje pre prvog objekta i nije vezana ni za jedan konkretno.

56) Чему је намењен и која су ограничења за писање преклопљеног оператора operator()?

Namenjen je uglavnom za određivanje vrednosti funkcija. Mora da bude nestatička metoda.

57) Да ли је механизам динамичког везивања ефикасан и зашто?

Mehanizam dinamičkog vezivanja je mehanizam kod kojeg se pokazivačem na osnovnu klasu pozovemo preklopljenu metodu iz njene izvedene klase. Ona je efikasna jer nije neophodno unapred znati koji je objekat predan. To je poenta polimorfizma.

58) Да ли је дозвољено да први аргумент функције нема подразумевану вредност, а други да је има и зашто?

Dozvoljeno je (i mora) jer prilikom pozivanja prevodilac povezuje argumente sa leva na desno, i u suprotnom ne bi moglo da se odredi šta je dodeljena a, a šta podrazumevana vrednost.

59) Како се може иницијализовати атрибут који је референца (упућивач) у некој класи?

Može se inicijalizovati pozivanjem inicijalizatora pri deklaraciji klase.

```
Class A
{
int &p;
public:
A (int k) : p(k) {}
};
```

60) Да ли се низ објеката изведене класе може пренети као аргумент преко показивача на објекат основне класе? Зашто?

Ne, jer količina memorije koju zauzimaju osnovna i izvedena klasa nisu iste, pa pristup nekom elementu pomeranjem pokazivača aritmetičkim operacijama (+ i -) neće biti moguć.

61) Ако је: class O {}; class I: public O{virtual m(){}}; … O* po=new I; који је резултат израза: typeid(*po)==typeid(I) ? Зашто?

False, jer je *po pokazvač na nepolimorfnu klasu O.

62) Шта означава релација пријатељства између две класе и да ли је она симетрична релација?

Relacija prijateljstva između klasa podrazumeva promenu prava pristupa, tako da klasa koja postaje prijateljska dobija pristup svim elementima klase kojoj postaje prijatelj.
Nije simetrična.

63) Да ли се у наредби A a=5; позива operator=(int) или конструктор A(int) и зашто?

Poziva se konstruktor A(int), jer se ovom naredbom objekat kreira i inicijalizuje, a ne dodeljuje se vrednost već stvorenom objektu.

64) Како се у изведеној класи може иницијализовати заштићена референца из основне класе? Навести пример.

Korišćenjem inicijalizacije. Primer bio.

65) Ако важи class I:O{}, коју вредност ће имати аргумент x по повратку из метода m: void m(int&x){x=0;try{x=1;throw new O;x=2;}catch(I*i){x=3;}x=4;}

Program se ne izvršava uspešno, a poslednja vrednost koju x dobija je 1.

66) Зашто се (по правилу) дефиниције класа на језику C++ пишу у датотекама-заглављима (*.h)?

Da bi bile dostupne svim fajlovima u projektu, uključivanjem datoteke pomoću naredbe #include“**.h“

67) Да ли су дозвољени и зашто (1) конструктори апстрактних класа и (2) променљиве типа показивача или упућивача (референце) на апстрактну класу?

(1) Jesu (2) Jesu

68) Која је улога итератора у STL–у, које врсте итератора постоје и шта омогућавају те врсте?

Uloga iteratora u STLu je pristup svim elementima zbirke.
Vrste iteratora: ulazni, izlazni, jednosmerni, dvosmerni, sa proizvoljnim pristupom. Omogućavaju: ulazni: čitanje elementa i kretanje u napred. Izlazni: menjanje vrednosti tekućeg elementa i kret. Unapred. Jednosmerni: osobine ulaznih i izlaznih. Dvosmerni osobine jednosmernog + kretanje unazad. Sa proiz pristupom: sve što i dvosmerni + pristup proizvoljnom elementu.

69) Да ли се препоручује да атрибути класе буду јавни и зашто?

Ne preporučuje se, jer one ne treba da budu dostupne ostatku programa, a ako su potrebne pišu se fje za dohvatanje atributa.

70) Колико пута се позива конструктор основне класе код виртуелног извођења у "дијамант структури"?

Jednom.

71) Да ли шаблонска класа може бити апстрактна и да ли функција може бити аргумент шаблона?

Šablonska klasa može biti apstraktna. Argument šablona može biti funkcija.

72) Шта је лвредност? Под којим условима је резултат тернарног оператора лвредност?

Lvrednost je vrednost koja postoji u memoriji i može joj se dohvatiti adresa, i može da stoji sa leve strane operatora dodele (=)
Rezultat ternarnog operatora je lvrednost akko je operand koji se vraća lvrednost.

73) Да ли су спољашња и угнежђена класа узајамно пријатељске? Како се изван спољашње класе приступа статичком (заједничком) члану угнежђене?

Nisu. Takvom elementu se izvan spoljašnje klase pristupa sa dva razrešenja dosega: Spoljašnja::Unutrašnja::Element.

74) Које ограничење важи при задавању подразумеваних вредности аргумената функција? Навести примере исправно и неисправно задатих подразумеваних вредности.

Parametri sa podrazumevanim vrednostima moraju biti navedeni desno od ostalih parametara.
Pr: ispravno: void f(int x, int y=0) neispravno void f(int x, int y=0, int z)

75) Да ли се оператор -> преклапа као бинарни или унарни оператор, да ли може да се преклопи као статичка метода и ког типа треба да буде резултат те операторске функције?

Preklapa se kao unarni operator. Ne može da se preklopi kao statička metoda. Rezultat funkcije je pokazivač na klasu koja sadrži element koji je predat fji: primer: klasa->ob se prevodi kao (klasa.operator->())->ob, sada je rezultat pokazivač na tip od ob

76) У чему је основна разлика између вишеструког извођења и извођења у више корака?

Osnovna razlika je što kod višestrukog izvođenja klasu izvodimo iz više osnovnih, a kod ovog drugog se iz izvedene klase izvodi nova klasa i tako dalje... više koraka:
class A{};
class B: class A{};
class C: class B{};
višestruko: class C: class B, class A {};

77) Шта је функцијска класа и да ли она може бити аргумент шаблона? Навести по један пример функцијских класа које реализују бинарне аритметичке, односно релационе операције из библиотеке STL (само назив).

Funkcijska klasa je klasa koja ima definisan operator() sa 1 i 2 operanda i primarna namena joj je realizacija unarnih i binarnih operacija (logičkih, aritmetičkih, relacionih)
Ne može.
Primeri: plus – binarna aritmetička
Equal_to – binarna relaciona

78) Која је разлика између података дефинисаних са и без модификатора static?

Podaci definisani sa modifikatorom *static* imaju produženi doseg, do kraja datoteke u kojoj su definisani.

79) Да ли се наслеђују и каква је подразумевана функционалност конструктора, деструктора и оператора = произвољне изведене класе?

Ne nasleđuju se, konstruktor stvara i inicijalizuje prazan objekat (svi elementi imaju svoje podrazumevane vrednosti), destruktor ima prazno telo, operator= prepisuje elemente polje po polje po redosledu definisanja u klasi.

80) Објаснити разлику између наредбе throw <izraz>; и клаузуле throw (<niz_id>).

Naredba throw „baca“ izuzetak i prijavljuje grešku, prekida try blok itd... dok klauzula throw, koja se nalazi u deklaraciji funkcije, definiše moguće tipove izuzetaka koje fja može da baci.

81) Навести начине употребе имена из неког простора имена.

Recimo da je definisano namespace A{ int x}
Može se upotrebiti na sledeće načine
A::x=3;
Using A::x;
x=3;
using namespace A;
x=3;

82) На коју меморијску локацију (i или p) показује &r ако је: int i=0,*p=&i,&r=p;?

Ni na jednu, jer je r definisano kao referenca na int, a p je tipa int*;

83) Дефинисати досег (област важења) идентификатора чланова класе и како се он може проширити.

Doseg identifikatora članova klase je deo programa u kojima se identifikator može pozivati i koristiti. On se može proširiti dodavanjem modifikatora static.

84) Шта је операнд, а шта резултат оператора typeid?

Operand ovog operatora može biti objekat nekog standardnog ili klasnog tipa, pokazivač, referenca, identifikator tipa itd... Rezultat je objekat klase type_info koji sadrži informacije o datom tipu.

85) Да ли у класи изузетка треба да постоји јавни конструктор копије и зашто?

Da, mora da postoji jer operator throw uvek kreira kopiju objekta klase izuzetaka.

86) Шта је намена итератора из стандардне библиотеке и шта означава тип reverse_iterator?

Namena je pristup svim elementima zbirke iz STL, a tip reverse_iterator označava takav iterator koji može da se kreće od poslednjeg ka prvom elementu promenljive zbirke.

87) Ако се у некој класи X дефинише operator()(int), како се за X x; int i; преводи x(i)?

Prevodi se kao x.operator()(i);

88) Да ли тип аргумента конструктора неке класе може бити: (1) сама та класа, (2) показивач на ту класу, (3) упућивач (референца) на ту класу?

(1) Ne (2) (3) može

89) Да ли је позив методе са динамичким везивањем (полиморфне методе) ефикаснији од позива методе са статичким везивањем и зашто?

Jeste. Jer se može kasnije, prilikom izvršavanja programa menjati njegov tok po potrebi i dinamički birati koje metode treba da se koriste.

90) Којим редоследом треба навести рутине за обраду (*handlers*) изузетака основне класе и класе изведене из те основне класе? Објаснити разлог.

Handler izvedene klase uvek mora biti naveden pre osnovne jer se pokazivač na izvedenu može implicitno konvertovati u pokazivač na osnovnu.

91) Да ли се аутоматско генерисање функције из шаблона врши при позивању дате функције у време извршења или у време превођења? Образложити одговор.

Prevođenja. Statički je.

92) Која је улога оператора за "реинтерпретирајућу конверзију типа"? Навести пример.

Reinterpret cast tumači bitsku reprezentaciju elementa na drugačiji način.
Float f;
Reinterpret_cast<int>(f)

93) Које су карактеристике аутоматски генерисаног подразумеваног конструктора? Када он постоји?

Karakteristika mu je da inicijalizuje sve atribute klase na podrazumevane vrednosti, a postoji kada korisnik(programer) ne navede ni jedan drugi konstruktor.

94) Како функција може да специфицира листу могућих изузетака и шта се догађа ако баци изузетак који није специфицирала?

Koristi se klauzula throw (svi mogući tipovi grešaka). Ukoliko se pojavi nespecificiran izuzetak poziva se fja unexpected() koja poziva abort(), koja vraća kontrolu operativnom sistemu.

95) Уколико се нека виртуелна основна класа O наслеђује вишеструко у класи X, колико пута ће се извршити њен конструктор при креирању објекта класе X?

Jednom.

96) Да ли је могуће имати референцу (упућивач) на показивач и показивач на референцу? Зашто?

* na & ne jer referenca nije lvrednost, nema adresu i ne može se na nju pokazati.
& na * da, svaki pokazivač je standardni tip koji se može referencirati.

97) *На које начине може да се позове функција operator()() за објекат x класе X?*

Može se pozvati kao x() ili kao x.operator()()

98) Шта значе појмови: досег и видљивост имена? Разлику појмова показати на примеру.

Doseg označava delove programa u kojima se može koristiti/pozivati, a vidljivost imena označava vreme kada posmatrani element postoji.

99) Која је намена универзалног руковаоца (*handler*) изузецима и на ком месту се наводи?

Namena univerzalnog hendlera (catch(...){}) jeste hvatanje izuzetaka koje nije obradio ni jedan prethodni hendler, bez obzira na to kog je tipa. Navodi se na kraju.

100) Које врсте аргумената се могу појавити у шаблону класе и која им је намена?

Standardni tipovi, klase, i konstanta bilo kog tipa. Namena im je omogućavanje stvaranja različitih klasa za različite tipove ili neke vrednosti konstanti.

101) *Да ли су дозвољене и зашто променљиве типа показивача или упућивача (референце) на апстрактну класу?*

Jesu, jer će one pokazivati na objekte konkretnih izvedenih klasa.

102) Који проблеми везани за динамичку алокацију меморије се најчешће појављују у C++ програмима?

Neuspela alokacija memorije. I još nešto??

103) Када су глобалне пријатељске функције погодније од функција чланица (метода)?

Pogodnije su npr kod binarnih operacija gde su oba operanda tipa klase koju posmatramo, pa su prirodnije.
104) Да ли се може написати апстрактна класа која нема ни једну апстрактну функцију и зашто?

Može se napraviti apstraktna klasa bez eksplicitnog navođenja apstraktne metode, tako što je izvedemo iz apstraktne klase, i ne redefinišemo čistu virtuelnu metodu.
Ipak, ova izvedena klasa će u sebi sadržati tu nasleđenu funkciju.

105) Написати произвољан једноставан пример шаблонске класе и пример њеног коришћења.

```
Class Tartalja{};
Template<class T> // isto kao typename
Class Klaslo{};
Klaslo<Tartalja> tegla;
```

106) Да ли се у време извршавања може променити тип елемента неке генеричке (шаблонске) збирке елемената? Зашто?

Ne, jer je mehanizam formiranja šablonskih klasa statički i dešava se za vreme prevođenja programa.