1. Ką abstrahuoja procedūrinio programavimo kalbos? Ką abstrahuoja objektinio programavimo kalbos?
2. Abstraktus duomenų tipas (ADT). Klasė. Objektas.
3. Kas yra programavimo paradigma? 5-ios Smalltalk charakteristikos, kurias suformulavo Alan Kay.
4. C++ programos vykdomojo failo sukūrimo etapai: preprocesorius, kompiliavimas, ryšių redagavimas (linking).
5. Išskaidyta programa. Išskaidytas kompiliavimas. **make** programa.
6. Kintamojo deklaravimas ir apibrėžimas. Funkcijos deklaravimas ir apibrėžimas.
7. Biblioteka. Antraštės (*header*) failas.
8. Dinaminės atminties išskyrimas. Operatoriai **new** ir **delete**.
9. Tipiška C biblioteka: struktūra ir su ja dirbančios funkcijos. Struktūros C kalboje trūkumai. C++ struktūra. Raktinis žodis **this**.
10. Inkapsuliacija *(encapsulation).* (Pastaba: Inkapsuliacijos terminas yra naudojamas pavadinti dvi skirtingas sąvokas. Kokios jos yra, ir kokia yra naudojama šiame kurse).
11. Matomumo kontrolė. **private, protected, public.**
12. Draugai (raktinis žodis **friend**).
13. Klasė ir C++ struktūra. Panašumai ir skirtumai.
14. Inicializacija *(initialization).*
15. Konstruktorius. Konstruktorius pagal nutylėjimą.
16. Destruktorius.
17. Objekto gyvavimo sritis *(scope).*
18. Objektų kūrimas ir sąlyginis programos vykdymas.
19. Funkcijų perkrovimas.
20. Funkcijos argumentai pagal nutylėjimą.
21. Kada funkcijas reikia perkrauti, o kada – naudoti argumentus pagal nutylėjimą.
22. Vardų erdvės *(namespace).* Kam reikia vardų erdvių.
23. Vienos vardų ervdės apibrėžimas skirtinguose failuose.
24. Alternatyvus vardų erdvės vardas *(alias).*
25. Vardų erdvė be pavadinimo.
26. Vardo iš vardų erdvės pasiekimas naudojant srities apsprendimo *(scope)* operatorių **::**
27. **using** direktyva.
28. Vardų iš vardų edvės, atvertos su **using** direktyva, uždengimas vardais, apibrėžtais einamojoje srityje.
29. Vardų iš vardų erdvių kolizija. **using** deklaracija.
30. Vardų erdvių atvėrimo rekomendacijos. **using** direktyva antraštės *(header)* faile.
31. Raktinis žodis **static** – kintamųjų/objektų sukūrimas duomenų segmente.
32. Statiniai kintamieji ir objektai.
33. Statiniai kintamieji ir statiniai objektai funkcijoje.
34. Statinių objektų konstruktorių ir destruktorių iškvietimas.
35. Klasės statiniai nariai kintamieji/objektai. Klasės statinės narės funkcijos.
36. Raktinis žodis **static** – vardų matomumo ryšių redagavimo metu reguliavimas.
37. Konstanta – reikšmė žinoma kompiliavimo metu.
38. Konstanta – kintamasis/objektas, kurio reikšmė po jo apibrėžimo negali būti pakeista.
39. Konstantų ryšių redagavimas pagal nutylėjimą.
40. Konstantinės rodyklės. Rodyklės reikšmė yra konstanta. Tai į ką rodo rodyklė yra konstanta.
41. Tipų suderinamumas priskiriant konstantą.
42. Funkcijos argumentas konstantinė reikšmė. Funkcijos argumentas adresas į konstantą.

43. Funkcijos grąžinama konstantinė reikšmė. Funkcijos grąžinamas adresas į konstantą.
44. Klasės narys kintamasis/objektas konstanta. Konstruktoriaus inicializatorių sąrašas. Klasės narys kintamasis/objektas statinė konstanta.
45. Klasės konstantinė narė funkcija.
46. Nuoroda *(reference).*
47. Nuoroda funkcijos argumentas. Nuoroda funkcijos grąžinama reikšmė.
48. Laikinasis objektas *(temporary)* . Laikinojo objekto perdavimas argumentu funkcijai.
49. Argumento perdavimas pagal nuorodą į konstantą - standartinis būdas perduoti argumentą pagal "reikšmę" C++ kalboje. Argumento perdavimas pagal nuorodą - standartinis būdas perduoti argumentą pagal "adresą" C++ kalboje.
50. Kada reikia perduoti pagal reikšmę. Kopijos konstruktorius. Kompiliatoriaus sukurtas kopijos konstruktorius. Programuotojo sukurtas kopijos konstruktorius.
51. Kopijos konstruktoriaus iškvietimas priskyrimo sakinyje.
52. Operatorius. Operandai. Operacija.
53. Vienvientis operatorius. Dvivietis operatorius.
54. Operatorių perkrovimas. Taisyklės.
55. Vienvietis ir dvivietis operatoriaus - globali funkcija.
56. Vienvietis ir dvivietis operatoriaus – klasės funkcija narė.
57. **++** ir **--** operatorių prefiksinės ir postfiksinės versijos perkrovimas.
58. Kada perkrauti operatorių kaip globalią funkciją ir kada perkrauti operatorių kaip klasės funkciją narę.
59. Kada perkraunamų operatorių operandas (argumentas) turėtų būti nuoroda ir kada – nuoroda į konstantą. Kada perkraunamas operatorius turi grąžinti (konstantinę) reikšmę ir kada – nuorodą, nuorodą į konstantą.
60. Kompozicija. Koks ryšys yra išreiškiamas kompozicija.
61. Paveldėjimas. Bazinė ir išvestinė klasės. Koks ryšys yra išreiškiamas paveldėjimu.
62. Funkcijos per-apibrėžimas *(redefinition).*
63. Sub-objektai. Sub-objektų sukūrimas konstruktoriaus inicializatorių sąraše.
64. Bazinės klasės (perkraunamų) funkcijų vardų paslėpimas išvestinėje klasėje.
65. Klasės sąsaja *(interface).*
66. Klasė, kuri yra bazinės klasės potipis (*subtype*).
67. Tipų *upcast*.
68. Saistymas (*binding*). Ankstyvasis saistymas ir vėlyvasis / dinaminis / vykdymo meto (*runtime*) saistymas. Kaip yra realizuotas dinaminis saistymas – VTABLE, VPTR. Raktinis žodis **virtual**.
69. Virtuali funkcija.
70. Funkcijų perrašymas (overwriting).
71. Polimorfizmas.
72. Dvi sąlygos, reikalingos, kad funkcijos kvietimas būtų polimorfinis.
73. Objekto apkarpymas (*object slicing*).
74. Švariai virtuali funkcija.
75. Abstrakti klasė. Švariai abstrakti klasė.
76. Švariai virtualios funkcijos apibrėžimas.
77. Virtualūs destruktoriai.
78. Virtualios funkcijos kvietimas iš klasės funkcijos narės, iš konstruktoriaus, iš destruktoriaus.
79. Konteineriai. Kam jų reikia.
80. Šabloninė klasė. Kompiliavimo meto konstantos šabloninėse klasėse.
81. Iteratorius.
82. Kas yra trikis *(exception).*
83. Trikio išmetimas. Kas yra atliekama vykdant **throw** sakinį.

84. Trikio gaudymas. **try-catch** blokas. Trikio tipo atitikimas gaudant trikį.
85. Trikiai konstruktoriuje. Kodėl juos reikia būtinai sugauti ir apdoroti.
86. Funkcijos/konstruktoriaus lygio **try-catch** blokas.
87. Iš standartinės C++ bibliotekos: **cin** ir **cout** objektų naudojimas; **string**, **ifstream**, **ofstream, logic_error, runtime_error** klasių naudojimas, **vector** šabloninės klasės naudojimas.