100 instuderingsfrågor inför slutprovet i Programmering 1

Slutprovet omfattar Lektioner **16**-**38** på webben m(o).
Kursboken: kap 7-10, sid 169-284, inkl.:
Övningar: kap 7, sid 200-204 kap 9, sid 267-269
kap 8, sid 220-222 kap 10, sid 284

Svar till alla frågor kan hittas genom lektyr i kursboken *Programmering 1 med C++*

Alla program- och sidoreferenser hänvisar till kursboken.

Kap 7 Funktioner, sid 169-204

Läs den introducerande texten om ***Funktionsbegreppet i programmering*** och fortsätt i kapitel 7.

1. Fornulera med egna ord funktionsbegreppet och förklara vad som är skillnaden och det gemensamma hos funktioner i matematiken och i programmeringen.
2. Varför kallas funktioner i programmering för *moduler* och vilka egenskaper har dessa moduler?
3. Nämn tre svar på frågan, varför man i programmering sysslar med funktioner.
4. Vad kallas indata som skickas till funktioner och hur betecknas utdata?
5. Hur många parametrar kan en funktion ha och hur många returvärden?
6. På vilket sätt är antalet returvärden från en funktion ”ett arv från matematiken”?
7. Vad innebär begreppet *modularisering* för programutveckling?
8. Vad är skillnaden mellan biblioteksprogram och program vi själva skriver?
9. Var någonstans placeras en funktions definition i ett C++ program?
10. Vilken funktion har vi hittills använt i alla våra program och varför?
11. Vilka beståndsdelar av en funktion gör att den kan användas även i andra program?
12. Varför måste funktioner *anropas*? Vad måste göras innan en funktion kan anropas?
13. Varför bidrar användningen av funktioner till återanvändning av kod?
14. Varför bidrar användningen av funktioner till strukturering av program?
15. Vilken naturlig strukturering av program bjuder på användningen av funktioner?
16. Vad gör `return`-satsen i en funktion förutom att skicka tillbaka funktionens värde?
17. Vad består *huvudet* av en funktion i C++ av?
18. Vad kallas den del av funktionshuvudet som innehåller parametrarna? Hur begränsas den?
19. Vad består *kroppen* av en funktion i C++ av och hur begränsas den?

20. Blir det en skillnad på koden om man placerar en funktions definition *före* eller *efter* `main()`?
21. Vilka tre händelser inträffar när man anropar en funktion?
22. Förklara skillnaden mellan *aktuella* och *formella parametrar*.
23. Formulera en allmän form på definition av en funktion med returvärde.
24. På vilket sätt skiljer sig anropet av en funktion med returvärde från en `void`-funktion?
25. Vad händer om man utelämnar `return`-satsen i en funktion?
26. Kan en `void`-funktion ha en `return`-sats?
27. Hur skiljer sig syntaxen mellan inkluderingen av en biblioteksfunktion från en egendefinierad funktion som man lagrat externt?
28. Vad är skillnaden mellan *definition* och *deklaration* av en funktion?
29. Vad menas med en funktions *signatur* och vilken betydelse har den?
30. Ge ett exempel på en *deklaration* av en funktion.
31. Varför är externlagring av funktioner av intresse i programutvecklingen?
32. Hur kan man göra för att få ut endast två decimaler vid utskrift av ett decimaltal?
33. Redogör för skillnaderna mellan *lokala* och *globala variabler*?
34. Vad menas med ett *block* i C++?
35. Vaför ger funtioner upphov till blockstruktur?
36. Kan man koda block i C++ utan funktioner*?*
37. Förklara med egna ord regeln för räckvidden av variabler.
38. Är en funktions parametrar globala eller lokala variabler i det program funktionen är definierad?
39. Vad är skillnaden mellan globala och lokala variabler vad gäller initieringen?
40. Vad är avgörande för att identifiera en variabel som global?
41. Vilken problematik kan uppstå vid deklaration av globala variabler?
42. Vad menas med *parametrisering av globala variabler* och vilken relevans har den?
43. Kan man ha i ett program samma namn för lokala som för globala variabler? Om nej, varför? Om ja, vad händer då?
44. Redogör för konceptet *överskuggning* av variabler.
45. Vad gör *räckviddsoperatorn* i C++ och hur kodas den?

Kap 8 Arrays och vektorer, sid 205-222

46. Kan man skapa en array vars element är av olika datatyper?
47. Kan man skapa en array vars element är arrays?
48. Vad innebär begreppet *index* hos arrays?
49. Vad måste man se upp med vid numreringen av en arrays element?
50. Vilket verktyg i C++ kontrollerar indexgränserna hos en array?
51. Vad säger *indexregeln*?

52. Vad är skillnaden mellan vanliga variabler och arrayelement vad gäller initieringen?
53. Nämn två olika betydelser av hakparenteserna **[ ]**. Hur kan man skilja mellan dessa betydelser?
54. Vad innebär en arrays *initieringslista* och hur ser den ut. Nämn ett exempel.
55. Nämn två nackdelar av array.
56. Redogör för skillnaderna mellan en array och en vektor?
57. Vilket bibliotek måste man inkludera i sitt program för arr kunna använda vektorer?
58. Skriv ett exempel på vektorns initieringslista.
59. Kan man skapa en vektor vars element är av olika datatyper?
60. Vad måste man tänka på när man vill skapa en sträng med hjälp av en array av **`char`**?
61. Vad innebär koden **`NULL`** i C++?
62. Vilken funktion måste man använda om man vill mata in en sträng som innehåller mellanslag?
63. Vilka två variabler måste kopplas ihop när man använder array i en **`for`**-sats?

Kap 9 Klasser, sid 223-269

64. Vad är den traditionella, procedurala synen på programmering som rådde på 60- och 70-talet?
65. Vad är den objektorienterade synen på programmering som kom upp på 80-talet?
66. Mellan vilka två programmeringsspråk går historiskt skiljelinjen mellan procedural och objektorienterad programmering? När ungefär inträffade övergången?
67. Vad var anledningen till paradigmskiftet inom programutveckling?
68. Vilka för- och nackdelar har enligt din åsikt den procedurala synen på programmering?
69. Vilka för- och nackdelar har enligt din åsikt den objektorienterade synen på programmering?
70. Vad menas med *paradigmskifte* i programmeringens historia?
71. Mellan vilka två programmeringsspråk går historiskt skiljelinjen mellan procedural och objektorienterad programmering? När ungefär inträffade övergången?
72. Vad var anledningen till paradigmskiftet inom programutveckling?
73. Vilka för- och nackdelar har enligt din åsikt den *procedurala* synen på programmering?
74. Vilka för- och nackdelar har enligt din åsikt den *objektorienterade* synen på programmering?
75. Är det korrekt att pepparkakor är *klasser* och pepparkaksformen *objekt*?
76. Kan man via *abstraktion* komma från objekt till klass eller är det tvärtom?
77. Om pennor är objekt var kan man hitta klassen *penna*?
78. Av vilka två huvudingredienser består en klass i regel?
79. Anta att *Tal* är en klass. Är *addition()* en metod eller en datamedlem i klassen *Tal*?
80. Anta att *Bil* är en klass. Är *Motor* en metod eller en datamedlem i klassen *Bil*?
81. Vad är skillnaden mellan *funktioner* och *metoder* i C++?
82. Vilka är den objektorienterade programmeringens tre hörnstenar?

83. Vad innebär modularisering på klassnivå?

Kap 10 Filhantering, sid 270-284

84. Vad är motivationen för filhantering i programmering?
85. Vilka två operationer står i centrum för filhantering?
86. Vilket bibliotek behövs för att kunna använda filhanteringsklasser i ett C++ program?
87. Vilken klass ur biblioteket i frågan ovan behövs för att skriva från ett C++ program till filer?
88. När och var någonstans i datorns fil- och mappsystem skapas filen **`Textfil.txt`** med satsen:

```
ofstream fileForWrite("Textfil.txt");
```

89. Vad händer om du exekverar satsen i frågan ovan om filen **`Textfil.txt`** redan finns?
90. Vilken del av satsen i fråga 5 skapar filhanteringsobjektet och vad är referensen till objektet?
91. Skriv kod som behövs för att skriva texten **`"Hallo!"`** till filen som skapas i fråga 108.
92. Skriv kod som stänger filen **`Textfil.txt`**. Vad händer när filen stängs efter användning?
93. Vilken klass behövs för att läsa från filer till ett C++ program?
94. Vad händer i satsen:

```
ifstream fileForRead("Textfil.txt");
```

95. Vad heter datamedlemmen som representerar filslutstecknet i i klassen **`ifstream`**?
96. Av vilken datatyp är filslutstecknet i klassen **`ifstream`**?
97. Skriv kod som läser allt innehåll från filen som nämns i fråga 94.
98. Modifiera satsen i fråga 94 för att lägga till text till filen **`Textfil.txt`** utan att det gamla innehållet raderas.
99. Modifiera funktionen **`randPasswd()`** (sid 277) genom att implementera en ny lösenord-policy för att generera slumplösenord: 3 gemener, 2 versaler och 2 specialtecken. Inkludera tecknen ? och @ i versalerna, för att de är ”grannar” i ASCII-tabellen.
100. Testa den nya policyn i frågan ovan för att skriva ut de nya slumplösenorden samt tillhörande användarnamn till en fil.