Datorlaborationer, Programmering i C++ (EDAF30)

Datorlaborationerna ger exempel på tillämpningar av det material som behandlas under kursen.

- *Uppgifterna i laborationerna ska lösas i par om två.* I samband med anmälan till till laborationerna får du möjlighet att ange vem du vill samarbeta med.
- *Laborationerna är obligatoriska.* Det betyder att du måste bli godkänd på alla uppgifterna under ordinarie laborationstid. Om du skulle vara sjuk vid något laborationstillfälle så måste du anmäla detta till kursansvarig lärare före laborationen (epost-adress finns på kursens hemsida).

  Om du varit sjuk bör du göra uppgiften på egen hand och redovisa den under påföljande laborationstillfälle. Det kommer också att anordnas en uppsamlingslaboration i slutet av kursen. Detta tillfälle erbjuds dock bara till dem som haft giltigt skäl för frånvaro på någon laboration eller som varit närvarande vid samtliga laborationer men, trots rimliga förberedelser, inte hunnit bli färdiga.
- *Laborationerna kräver en hel del förberedelser.* I början av varje laboration finns anvisningar om förberedelser under rubrikerna Läsanvisningar och Förberedelser. Läsanvisningarna anger vilka avsnitt i läroboken som ska läsas. Under rubriken Förberedelser anges vilka av laborationsuppgifterna som *ska* lösas före laborationstillfället. Du ska också ha läst igenom de övriga uppgifterna och gärna försökt lösa dem. Det är inget krav att att du kommer med helt färdiga lösningar. Men det är ditt och din laborationspartners ansvar att ha förberett er så att ni bedömer att ni hinner bli klara under laborationen. Ni får naturligtvis under dessa förberedelser gärna kontakta kursansvarig lärare om ni stöter på svårigheter.
- *Utveclingsmiljöer.* På skolans datorer finns CodeBlocks och Visual C++ 2010 Express installerat. Om du vill jobba med laborationen hemma och inte redan laddat ner någon utvecklingsmiljö för C++ så finns anvisningar för detta på kursens hemsida. Några uppgifter använder det grafiska gränssnittet *wxWidgets* som också kan laddas ner via hemsidan.
- Du måste för varje laboration se till att laborationsledaren noterar dig som godkänd på listan på nästa sida.

Om du hittar någonting i uppgifterna eller andra anvisningar som är felaktigt eller oklart så meddela gärna detta till kursansvarig lärare.

Innehåll

Programmering i C++, godkända laborationsuppgifter

Skriv ditt namn och din namnteckning nedan:

Namn: .....

Namnteckning: .....

Godkänd laboration	Datum	Laborationsledarens namnteckning
1		
2		
3		
4		
5		
6		

Laboration 1

Mål: Du ska på de områden som behandlats under de första föreläsningarna: de grundläggande delarna av C++ samt tecken och texter.

Läsanvisningar

Läs igenom följande avsnitt i boken: 2.1–2.10, 3.1–3.4. Läs också på föreläsningsbilderna från föreläsning 1 och 2, som finns på kursens hemsida.

Förberedelser

Lös uppgifterna U1-U3 och U6-U8 nedan under rubriken ”Uppgifter”. Läs igenom uppgifterna U4-U5 och U9-U10. Uppgift U9 är frivillig.

Uppgifter

H1. Skriv om följande uttryck med if-satser:

```
x = (a > b) ? a : (b > c) ? b : c ;
```

Lägg till kod så att `x` innehåller maxvärdet av `a`, `b` och `c`. Testa det omskrivna uttrycket i följande kodavsnitt:

```
for (int a = 1; a <= 3; a++) {
  for (int b = 1; b <= 3; b++) {
     for (int c = 1; c <= 3; c++) {
         int x = 0;
         int m = a; if (b > m) m = b; if (c > m) m = c;

         // lägg in omskriven kod att testa här
         // x = (a > b) ? a : (b > c) ? b : c ;

         cout << "a b c x = " << a << " " << b << " " << c << " " << x;
         if (x != m) cout << " Fel: max = " << m;
         cout << endl;
     }
  }
}
```

U2. Skriv ett program som kontrollerar om och hur mycket man vunnit i ett lotteri av typ Joker. Programmet kan antingen själv generera det sexsiffriga vinstnumret (slumptal) eller också låta användaren mata in det först. Man ska sedan kunna mata in flera lottnummer för att kontrollera eventuell vinst. Vinsten är 10 kr om första (entals-) siffran överensstämmer, 100 kr om de två första överensstämmer osv. Det finns även en extra utdelning på 250 kr om de tre första siffrorna förekommer i samma ordning någon annanstans i vinstnumret (står de först vinner man 1000 kr).

U3. I vissa olympiska gymnastikgrenar beräknas den tävlandes resultat som medeltalet av erhållna domarpoäng efter att man har tagit bort den högsta och lägsta poängen. Skriv ett program som läser in domarpoängerna (3 eller fler) och beräknar resultatet enligt ovan.

U4. Ändra uppgiften ovan så att domarpoängen läses in från en fil och gör en lämplig fil. OBS! Decimalpunkt.

U5. Skriv ett program som slumpar fram en av tre möjliga utfall och skriv ut texten STEN, SAX eller PÅSE beroende på utfallet. Lägg sedan till att man får mata in ett av alternativen först och att programmet talar om vem som vann.
(STEN vinner över SAX, SAX vinner över PÅSE och PÅSE vinner över STEN)

U6. Skriv ett program som läser in en sträng och sen skriver ut den baklänges.

U7. Skriv ett program som avgör om ett angivet bilnummer är giltigt. Utgå från den enkla formen med tre bokstäver följt av tre siffror (t.ex. HEJ345).

U8. Det s.k. Caesarchiffret går ut på att rotera alfabetet ett visst antal positioner och på så sätt obegripliggöra ett meddelande. Skriv ett program som läser in ett meddelande i form av en sträng på högst 80 tecken. Meddelandet krypteras genom att förskjuta varje bokstav 13 positioner i alfabetet (engelskt alfabet med 26 bokstäver). Efter 'Z' börjar alfabetet om igen från 'A'. Alla gemener (små bokstäver") i meddelandet görs först om till versaler (stora bokstäver") innan själva krypteringen genomförs.

U9. Modifiera föregående uppgift så att krypteringen görs genom en godtycklig permutation av bokstäverna A-Z. Enklast är att lagra det permuterade alfabetet i en sträng, t.ex. "QWERTYUIOPASDFGHJKLZXCVBNM".

U10. (bokens övning 3.9): De romerska siffrorna anges med bokstäverna I, V ,X ,L ,C ,D och M som står för 1, 5, 10, 50, 100, 500 respektive 1000. Deklarera först en string-variabel som innehåller de romerska siffrorna och sedan en tabell (ett fält) som kan användas för att översätta en romersk siffra till ett vanligt heltal (t.ex. L till 50). Skriv sedan ett program som läser in ett romerskt tal till en string-variabel och som översätter det romerska talet till ett vanligt heltal. Om användaren t.ex. skriver MCMXLIX skall programmet skriva ut 1949. Programmet skall ge en felutskrift om det inmatade romerska talet är felaktigt. I ett romerskt tal gäller att om en romersk siffra P står omedelbart till vänster om en annan romersk siffra Q och om P betecknar ett mindre tal än Q, så skall värdet av P subtraheras från det totala talet (LIX betyder t.ex. 59), annars skall P adderas till det totala talet (LXI betyder 61).

Laboration 2

Mål: Du ska träna på att konstruera och anropa egna funktioner i C++. Du ska också träna på användning av pekare i C++.

Läsanvisningar

Läs kap 4–5 i läroboken. Läs också föreläsningsbilder från de föreläsningar, som behandlar motsvarande avsnitt. Dessa finns på kursens hemsida.

Förberedelser

Läs igenom den inledande texten under rubriken ”Uppgifter” nedan och lös uppgift U1-U3, U5, U8, U10 och U12-U13. Läs igenom och sätt dig in i uppgifterna U4, U6-U7, U9, U11 och U14. Uppgifterna U3-U4 och U10 är frivilliga.

Uppgifter

I dessa övningar ingår att skriva en `main`-funktion för test av funktionerna.

U1. Skriv en funktion `delbar` som har 2 heltalsparametrar (`a` och `b`) och returnerar `true` om `a` är jämnt delbar med `b`, annars `false`.

U2. Skriv en funktion `kvadratkubtabell` som har en värdeparameter `n` av typen `int` och som skriver ut en tabell med heltalen från `1` till `n` samt deras kvadrater och kuber. Förse tabellen med lämplig rubrik.

U3. Det finns en standardfunktion, `rand` (deklareras i `<cstdlib>`), som returnerar s k pseudoslumptal, heltal i intervallet 0 till `RAND_MAX` (systemberoende).
Skriv en funktion, `randInt(min, max)`, som i stället returnerar ett heltal i intervallet `min..max` (låt funktionen utnyttja `rand` för att åstadkomma detta).

U4. Skriv en funktion `downcase` som tar in en C-sträng och ändrar alla versaler (STORA BOKSTÄVER) i strängen till gemener (små bokstäver).

U5. Ett polynom av fjärde graden kan skrivas
$$p(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$$
En funktion `poly` ska skrivas som beräknar polynomfunktionens värde för godtyckligt val av `x`. Polynomkoefficienterna läggs i en vektor som får utgöra en av inparametrarna till funktionen.

U6. Skriv en funktion som tar en heltalsvektor (`std::vector<int>`) som inparameter samt ökar varje heltal i vektorn med `1`. Använd inparametern som utparameter också samt utnyttja referensanrop.

U7. Skriv två olika varianter av en funktion som beräknar summan av kvadraterna på alla heltal från `1` till ett givet tal `n`. Den ena varianten skall vara iterativ och den andra rekursiv.

U8. Gör en egen version `mystrcat` av funktionen `strcat(*char s,const *char t)` som lägger till strängen `t` till slutet av strängen `s` (konkatenering).

U9. Skriv en funktion, `void strreverse(char *s)` , som vänder på innehållet i C-strängen `s`. Om man t ex anropar funktionen med ’’123’’ funktionen ge strängen ändras till ’’321’’.

U10. I C-biblioteket `<cstring>` finns funktionen `int strcmp(const char* s1, const char* s2);` Denna funktion jämför C-strängarna `s1` och `s2`. Resultatet är `< 0` om `s1` är mindre än `s2`, `0` om strängarna är lika och `> 0` om `s1` är större än `s2`. Jämförelsen sker tecken för tecken i ASCII-ordning. Exempelvis är `''Berit''` mindre än `''Bertil''` eftersom det fjärde tecknet skiljer sig och `'i'` är mindre än `'t'`. `''Bert''` är mindre än `''Bertil''`, eftersom första strängen är kortare än den andra (och de gemensamma tecknen är lika). Implementera `mystrcmp` med samma beteende som `strcomp`, utan att utnyttja några standardfunktioner.

U11. Skriv funktionen `void byt(int &a, int &b)` som byter plats på `a` och `b` om `a > b`. Använd sedan funktionen för att sortera en heltalsvektor `v` med `n` element enligt följande princip: Jämför `v[k]` med varje element med index `>k` och byt om `v[k]` är störst. Låt `k` gå från `0` till `n-1`.

U12. Skriv ett program som frågar efter ett meddelande (textsträng) och sen kodar om den till morsesignaler genom att skriva ut morsesymbolerna (med blanktecken emellan). Morsealfabetet ges av följande tabell:

```
A .-     B -...   C -.-.   D -..    E .      F ..-.
G --.    H ....   I ..     J .---   K -.-    L .-..
M --     N -.     O ---    P .--.   Q --.-   R .-.
S ...    T -      U ..-    V ...-   W .--    X -..-
Y -.--   Z --..   Å .--.-  Ä .-.-   Ö ---.
```

U13. Skriv ett program som slumpar ut talen 0-15 i en 4x4-matris och därefter skriver ut den. Exempelutskrift:

```
 3  7  9  8
11  6  1  4
 5  2 14 10
12  0 15 13
```

U14. Komplettera föregående uppgift så att man kan spela det s.k. 15-spelet. Talen 1-15 tolkas här som brickor som kan flyttas om det är en lucka intill (0). Eventuellt blir pariteten fel (t.ex. alla rätt utom två brickor som har bytt plats) och då saknar spelet lösning. Detta problem bortser vi emellertid ifrån vid slumpningen (i förra uppgiften) för enkelhets skull. Dialogexempel (med utgångsposition enligt ovan):

```
Vilken bricka ska flyttas? 12
  3  7  9  8
 11  6  1  4
  5  2 14 10
  0 12 15 13
```