Örebro universitet
Akademin för naturvetenskap och teknik
Thomas Padron-McCarthy (thomas.padron-mccarthy@oru.se)

Tentamen i

Programmering i språket C

för D1 m fl, även distanskursen

lördag 25 februari 2012

Gäller som tentamen för:
DT1016 Datateknik A, Programmering grundkurs, provkod 0100
DT1007 Datateknik A, Tillämpad datavetenskap, provkod 0410
DT1006 Datateknik A, Programmering C, distans, provkod 0100

Hjälpmedel:	Inga hjälpmedel.
Poängkrav:	Maximal poäng är 40. För godkänt betyg (3 respektive G) krävs 20 poäng.
Resultat och lösningar:	Meddelas via e-post eller på kursens hemsida, http://basen.oru.se/kurser/c/2011-2012-p2/, senast lördag 17 mars 2012.
Återlämning av tentor:	Efter att resultatet meddelats kan tentorna hämtas på universitetets centrala tentamensutlämning.
Examinator och jourhavande:	Thomas Padron-McCarthy, telefon 070-73 47 013.

- Skriv tydligt och klart. Lösningar som inte går att läsa kan naturligtvis inte ge några poäng. Oklara och tvetydiga formuleringar kommer att misstolkas.
- Skriv den personliga tentamenskoden på varje inlämnat blad. Skriv *inte* namn eller personnummer på bladen.
- Skriv bara på en sida av papperet. Använd inte röd skrift.
- Antaganden utöver de som står i uppgifterna måste anges.
- Skriv gärna förklaringar om hur du tänkt. Även ett svar som är fel kan ge poäng, om det finns med en förklaring som visar att huvudtankarna var rätt.

LYCKA TILL!

Prioritet och associativitet hos operatorerna i C

De viktigaste operatorerna:

Prioritet	Kategori	Operator	Associativitet
Högsta	Unära postfixoperatorer	(), [], ->, ., ++, --	vänster
	Unära prefixoperatorer	!, ++, --, +, -, *, &, sizeof, (*typ*)	höger
	Multiplikation mm	*, /, %	vänster
	Addition mm	+, -	vänster
	Jämförelser	<, <=, >=, >	vänster
	Likhetsjämförelser	==, !=	vänster
	Logiskt OCH	&&	vänster
	Logiskt ELLER	||	vänster
Lägsta	Tilldelning	=, +=, -=, *=, /=, %=	höger

Uppgift 1 (1 p)

Vilka värden har följande C-uttryck?

a) 1 + 2 * 3

b) 1 - 2 - 3

c) 1 - 2 / 3

d) 1 < 2 - 3

Uppgift 2 (1 p)

Variablerna **a**, **b** och **c** är av typen **int**, och variabeln **x** är av typen **float**. Vilka värden har variablerna efter att följande kod har körts?

```
a = 1; b = a + 1; c = b + 1; x = c + 1;
if (a < b) {
    while (b < c)
        ++b;
    x = x + c;
}
```

Uppgift 3 (3 p)

Skriv ett komplett C-program (med **#include** och allt) som läser in två heltal. Om bägge heltalen är lika med **17** ska programmet skriva **Sjutton!**. Annars ska programmet skriva ut de två heltalen, men i omvänd ordning, alltså med det sist inmatade talet först.

> I den här och alla andra uppgifter på tentan gäller:
> Normalt är felhantering en stor del av ett program. Vad ska till exempel hända om användaren skriver **Kalle** när hon egentligen borde mata in ett tal? Här behövs dock ingen felhantering, om så inte särskilt efterfrågas i uppgiften.

Uppgift 4 (2 p)

Skriv en funktion, **samma_inmatade_tal**, som tar ett heltal som argument, och som också låter användaren mata in ett heltal. Om dessa båda tal är lika, ska funktionen returnera ett sant värde. Annars ska funktionen returnera ett falskt värde.

Uppgift 5 (1 p)

Vi vill kontrollera om användaren matar in talet **42**. Skriv en **main**-funktion som anropar funktionen **samma_inmatade_tal** med **42** som argument, och om **samma_inmatade_tal** returnerar ett sant värde ska **main** skriva ut **Ja**. Annars ska **main** skriva ut **Nej**.

I den här och alla andra uppgifter på tentan gäller: Om du behöver använda något från en tidigare uppgift eller deluppgift, till exempel anropa en funktion som skrevs i den tidigare uppgiften, så behöver du inte skriva samma kod igen. Du får också göra uppgiften även om du inte gjort den tidigare uppgiften.

Uppgift 6 (6 p)

På en Lotto-bricka finns det 35 nummer, numrerade från 1 till 35. Man ska välja sju av dessa. Man får bara välja varje nummer en gång.

Skriv ett C-program som läser in en lottorad bestående av sju heltal, och talar om ifall det är en tillåten rad. Det som ska kontrolleras är:

- Är alla talen mellan 1 och 35?
- Står talen i nummerordning?
- Förekommer talen bara en gång var?

Man kan förutsätta att användaren alltid matar in sju riktiga heltal.

Ett exempel på en tillåten lottorad är **7 10 16 23 26 27 35**. Tre exempel på lottorader som inte är tillåtna är **7 10 16 23 26 27 36**, **7 10 16 23 26 27 27** och **7 10 16 23 26 35 27**.

Uppgift 7 (1 p)

När man ritar bilder med en dator använder man för det mesta pixelgrafik, där ytan som ska ritas upp delas in i bildpunkter, och varje bildpunkt har en färg och ljusstyrka som anges med hjälp av tre färgvärden: rött, grönt och blått. Man kan tänka sig att det sitter tre små lampor i varje bildpunkt: en röd lampa, en grön lampa och en blå lampa. Därför behövs det tre tal för att beskriva färgen på varje bildpunkt. Vi tänker oss att vi anger färgerna med flyttal som kan variera från **0** (helt släckt) till **1** (maximal ljusstyrka). Exempelvis betyder taltrippeln **0.5-0-0** en halvstarkt lysande röd färg, eftersom den "röda lampan" är halvtänd och de gröna och blå "lamporna" är släckta.

Skapa posttypen **Pixel**. Den innehåller tre flyttal, som anger de tre färgernas värden för den bildpunkten.

Uppgift 8 (1 p)

Definiera en variabel av typen **struct Pixel** och initiera den med den halvstarkt lysande röda färgen från uppgiften ovan.

Uppgift 9 (2 p)

Skriv en funktion som heter **equal_colours**, som jämför två **Pixel**-poster och returnerar ett sant värde om de har samma färg. I annat fall ska den returnera ett falskt värde.

Uppgift 10 (1 p)

När man ska rita upp saker behövs förstås en hel masssa bildpunkter. Skapa därför posttypen **Canvas** som ska användas för att beskriva en rityta som är 1000 gånger 1000 bildpunkter stor. (Det engelska ordet "canvas" betyder ungefär "målarduk".)

Använd först **#define** för att definiera makrona **CANVAS_WIDTH** och **CANVAS_HEIGHT**, som båda ska ha värdet **1000**. Definiera därefter posttypen **Canvas**, som är en post som innehåller en 1000x1000-matris av **Pixel**-poster.

Tips: En matris, eller tvådimensionell array, är helt enkelt en array av arrayer. Kom ihåg hur man skapar en array: **int a[17]** är en array av 17 stycken heltal. **int m[20][30]** är en 20x30-matris av heltal. På samma sätt som man kan komma åt plats 3 i arrayen **a** genom att skriva **a[3]**, kan man komma åt plats (5, 7) i matrisen **m** genom att skriva **m[5][7]**.

Uppgift 11 (3 p)

Skriv en funktion som heter **solid**, som fyller en hel **Canvas** med samma färg. Den ska ta två parametrar: dels en pekare till en **Canvas**-post, och dels en **Pixel**-post. Funktionen ska sätta alla bildpunkterna i ritytan så de får samma färg som den medskickade bildpunkten.

Uppgift 12 (3 p)

Skriv en funktion som heter **border**, som ritar en ram längs kanten av en **Canvas**. Den ska ta två parametrar: dels en pekare till en **Canvas**-post, och dels en **Pixel**-post. Funktionen ska sätta alla de yttersta bildpunkterna längs kanten på ritytan så de får samma färg som den medskickade bildpunkten.

Uppgift 13 (3 p)

Skriv en funktion som heter **equal_canvases**, som jämför två **Canvas**-poster och returnerar ett sant värde om de är lika, dvs om varje bildpunkt i den ena ritytan har samma färgvärde som bildpunkten på samma plats i den andra ritytan. I annat fall ska den returnera ett falskt värde. Använd funktionen **equal_colours** för att jämföra färgerna på bildpunkterna.

Uppgift 14 (2 p)

Skriv en **main**-funktion som har två lokala variabler av typen **struct Canvas**. Den ska börja med att först göra den ena ritytan enfärgad med hjälp av funktionen **solid**. Sedan ska den rita en ram med en annan färg på den ritytan med hjälp av funktionen **border**. Sedan ska den göra samma sak med den andra ritytan. Som avslutning ska den använda funktionen **equal_canvases** för att kontrollera att de båda ritytorna verkligen ser likadana ut. Om de *inte* gör det ska den skriva ut **Fel! Olika!**. Om de är lika ska inget skrivas ut.

Uppgift 15 (10 p)

a) (4p)
Skriv en funktion **save_canvas** som tar två parametrar: en **Canvas**-post (eller, om du vill, en pekare till den) och en textsträng. Funktionen ska öppna filen med det angivna namnet, skriva ritytans data på den i något lämpligt format, och sedan stänga filen. Om filen inte går att öppna för skrivning, ska ett felmeddelande skrivas ut, och programmet ska avslutas.

b) (4p)
Vi vill förstås också ha en funktion som kan läsa in ritytan från filen. Skriv den funktionen. Den ska heta **read_canvas**. Om filen inte går att öppna för läsning, ska ett felmeddelande skrivas ut, och programmet ska avslutas.

c) (2p)
Nu vill vi provköra funktionerna **save_canvas** och **read_canvas**. Skriv en **main**-funktion som först sparar en rityta på en fil med hjälp av **save_canvas**, och sen läser in den med **read_canvas**. Därefter ska den använda funktionen **equal_canvases** för att kontrollera att den inlästa ritytan är likadan som den vi sparade på filen. Om de *inte* lika ska den skriva ut **Fel! Olika!**. Om de är lika ska inget skrivas ut.
