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

Tentamen i

Programmering grundkurs och Programmering C

för D1 m fl, även distanskursen

lördag 6 mars 2010

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://www.aass.oru.se/~tpy/c/2009-2010-p2/, senast lördag 27 mars 2010.
Återlämning av tentor:	Efter att resultatet meddelats kan tentorna hämtas på universitetets centrala tentamensutlämning i L1506, måndag till torsdag kl 10-14.
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 uttryck?

a) 1 * 2 + 3 * 4

b) 1 * 2 - 3 - 4

c) 1 * 2 * 3 / 4

Uppgift 2 (1 p)

x, **y** och **z** är flyttalsvariabler av typen **double**. Vad händer när följande kod körs?

```
x = 1.0; y = 2.0; z = 3;
if (x < y + z)
    z = z * 2;
else
    z = z * 3;
while (x < y + 1) {
    z = z + 1;
}
```

Uppgift 3 (1 p)

Vi ska simulera ett 3-dimensionellt flipperspel med kulor av olika storlekar. Uppgifter om de olika kulorna ska lagras i poster ("structar") av typen **struct Kula**. En kul-post innehåller kulans **diameter** (i millimeter), dess **vikt** (i gram), och en position som bestäms av de tre koordinaterna **x**, **y** och **z** (som anges i millimeter). Allt är flyttal.

Definiera posttypen **struct Kula**.

Uppgift 4 (1 p)

Definiera en variabel av typen **struct Kula** och initiera den med data om en kula med diametern 30 mm som väger 83 gram och som befinner sig i positionen x = 0, y = 450, z = 0.

Uppgift 5 (2 p)

Vi vill kunna visa kulposternas innehåll på skärmen. Skriv en funktion som heter **visa_kula**, som skriver ut data om ett kula. Funktionen ska ta en kulpost (eller, om du vill, en pekare till den) som parameter. Så här kan utskriften se ut:

```
Diameter: 30.00
Vikt: 83.00
Position: x = 0.00, y = 450.00, z = 0.00
```

Uppgift 6 (2 p)

Skriv en funktion som heter **las_kula**, och som läser in data om ett kula. Funktionen ska skriva ut lämpliga ledtexter på standardutmatningen, och läsa in data från standardinmatningen (som normalt är kopplad till tangentbordet).

Du får själv välja om du vill att funktionshuvudet ska se ut så här:

```
struct Kula las_kula()
```

eller så här:

```
void las_kula(struct Kula *p)
```

Uppgift 7 (3 p)

Vi vill skriva en funktion som heter **krockat** som tar två kulposter som argument, och som returnerar ett sant värde om de två kulorna har krockat. Om de inte har krockat, ska den returnera ett falskt värde.

Kulornas koordinater anger deras mittpunkter. Om två kulor har koordinaterna **x1**, **y1**, **z1** respektive **x2**, **y2**, **z2**, anges avståndet **a** mellan de två kulornas mittpunkter av formeln:

$$a = \sqrt{(x1 - x2)^2 + (y1 - y2)^2 + (z1 - z2)^2}$$

Om det avståndet **a** är mindre än summan av kulornas radier, har kulorna krockat. (Radien är lika med halva diametern.)

Uppgift 8 (3 p)

Skriv en main-funktion som har två lokala variabler av typen **struct Kula**, och som läser in data om två kulor till dessa variabler med hjälp av funktionen **las_kula**. Därefter ska programmet tala om ifall dessa båda kulor har krockat eller inte. (Använd funktionen **krockat** för att kolla detta.)

Om du ska 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 9 (4 p)

Skriv ett C-program som läser in upp till 1000 kulor med hjälp av funktionen **las_kula**, och sen talar om de inlästa kulornas sammanlagda vikt.

Programmet ska avslutas genom att man matar in en kula med negativ vikt. (Den ska inte räknas med i den sammanlagda vikten.)

Uppgift 10 (4 p)

Skriv ett C-program som läser en textrad från standardinmatningen, och sedan skriver ut den baklänges. Raden kan vara högst 80 tecken lång (radslutstecknet ej medräknat). Om användaren matar in mer än 80 tecken, ska man få ett felmeddelande om detta.

Uppgift 11 (3 p)

Skriv en funktion som tar en array av flyttal som argument, samt ett tal som anger hur många flyttal som arrayen innehåller, och som returnerar det *näst* högsta talet i arrayen. Vi kan anta att alla tal i arrayen är olika, och att den består av minst två tal.

Uppgift 12 (3 p)

Skriv ett C-program som läser heltal från standardinmatningen, som avslutas när talet 0 matas in, och som då skriver ut de *två* högsta heltalen. Vi kan anta att alla inmatade tal kommer att vara olika, och att inmatningen består av minst två tal.

Det här programmet ska *inte* lagra de inmatade talen i en array eller liknande, utan det räcker att hela tiden hålla reda på de två tal som hittills är högst.

Uppgift 13 (4 p)

Skriv ett C-program som läser heltal från standardinmatningen, som avslutas när talet 0 matas in, och som då skriver ut alla inmatade tal *utom* de två högsta. Vi kan anta att alla inmatade tal kommer att vara olika, och att inmatningen består av minst två tal.

Programmet ska lagra de inmatade talen i en array, och kunna hantera åtminstone 45000 inmatade tal.

Uppgift 14 (8 p)

Skriv ett C-program som läser heltal från standardinmatningen, som avslutas när talet 0 matas in, och som då skriver ut alla inmatade tal *utom* de två högsta. Vi kan anta att alla inmatade tal kommer att vara olika, och att inmatningen består av minst två tal.

Det här programmet ska kunna hantera flera miljarder inmatade tal. Det är fler tal än som får plats att lagra i en array, och därför måste de lagras på en fil.
