Örebro universitet
Institutionen för teknik
Thomas Padron-McCarthy (Thomas.Padron-McCarthy@tech.oru.se)

Tentamen i

Programmering grundkurs och Programmering C

för D1 m fl, även distanskursen

lördag 31 maj 2008 kl 14:00 - 18:00

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
PRG045 Programmering C, provkod 0110 (och 0100, och 0101)
TDD121 Tillämpad datavetenskap A, provkod 0700 (och 0400)
TDP118 Programmering C, distans, provkod 0110 (och 0100)

Hjälpmedel:	Inga hjälpmedel.
Poängkrav:	Maximal poäng är 32. För godkänt betyg (3 respektive G) krävs 16 poäng.
Resultat och lösningar:	Meddelas via e-post eller på distanskursens hemsida, http://basen.oru.se/kurser/c/2007-2008-p34-distans/, senast lördag 21 juni 2008.
Återlämning av tentor:	Efter att resultatet meddelats kan tentorna hämtas på institutionen. Man kan också få sin rättade tenta hemskickad.
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 - 3.0 / 4

Uppgift 2 (1 p)

i och **k** är heltalsvariabler, och **x** är en flyttalsvariabel. Ange värdet på i, k och x då följande kod har körts.

```
i = 0;
x = 3.0;
while (i < 5) {
    k = i;
    if (i == 4)
        x = x / 4;
    i = i + 2;
}
```

Uppgift 3 (1 p)

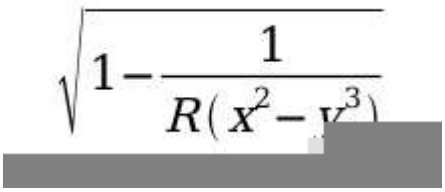
x och **y** är postvariabler av typen **struct Igelkott**. Skriv de kodrader som behövs för att byta plats på innehållet i dessa två variabler.

Uppgift 4 (1 p)

Skriv en funktion som heter **plus**, som tar två flyttal som argument, och som returnerar summan av dem.

Uppgift 5 (4 p)

Vi vill kunna beräkna följande uttryck:



Skriv därför ett komplett C-program som först läser in värdena på x, y och R, beräknar uttrycket, och till sist skriver ut uttryckets värde. Beräkningarna ska ske med flyttal.

Om nämnaren i divisionen (dvs deluttrycket under divisionsstrecket) är noll, eller om deluttrycket inuti rottecknet är mindre än noll, går uttrycket inte att beräkna. I så fall ska programmet inte försöka beräkna uttrycket, utan det ska i stället skriva ut ett informativt och rättvisande felmeddelande om saken.

Uppgift 6 (1 p)

Nu ska vi jobba vidare med posttypen **struct Igelkott**, som vi redan sett i uppgift 3 ovan.

En sådan post används för att lagra data om en igelkott. En igelkott har ett *namn*, en *vikt*, och ett *antal taggar*. Namnet kan vara högst tio tecken långt, till exempel **Karl-Oskar**. Vikten ska lagra som ett flyttal, medan antalet taggar ska vara ett heltal.

Definiera posttypen **struct Igelkott**.

Uppgift 7 (1 p)

Definiera en variabel av typen **struct Igelkott** och initiera den med data om igelkotten **Pilt**, som väger 1.3 kilo och har 7000 taggar.

Uppgift 8 (2 p)

Vi vill kunna visa igelkottposternas innehåll på skärmen. Skriv en funktion som heter **visa_igelkott**, som skriver ut ett igelkott (som i uppgifterna ovan) på skärmen. Funktionen ska ta igelkottposten som parameter. Exempel på hur en utskrift skulle kunna se ut:

```
Igelkott: Pilt
Vikt: 1.3 kg
Taggar: 7000
```

Uppgift 9 (2 p)

Skriv en funktion som heter **las_igelkott**, och som läser in data om en igelkott. 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 Igelkott las_igelkott()
```

eller så här:

```
void las_igelkott(struct Igelkott *p)
```

Uppgift 10 (2 p)

Skriv en main-funktion som har två lokala variabler av typen **struct Igelkott**, och som läser in data om två igelkottar till dessa variabler med hjälp av funktionen **las_igelkott**. Avslutningsvis ska den använda funktionen **visa_igelkott** för att skriva ut data om båda igelkottarna.

I den här och alla andra uppgifter på tentan gäller: 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 11 (4 p)

Skriv ett C-program som låter användaren mata in igelkottar, och som sparar dem på en fil. Det ska läsa igelkottar ända tills användaren avslutar genom att mata in en igelkott som väger 0 kilo. Programmet ska anropa funktionen **las_igelkott**.

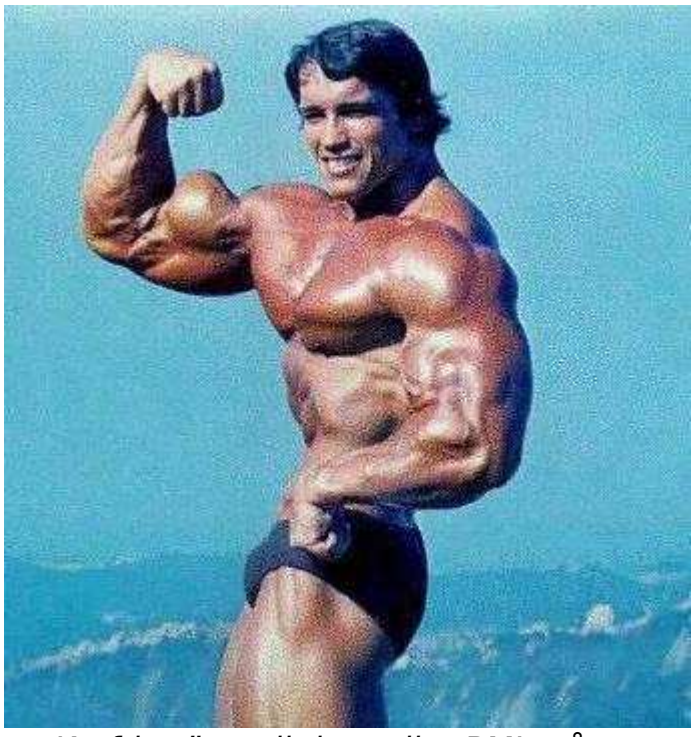
Uppgift 12 (4 p)

Skriv ett C-program som först frågar efter ett namn, och sen läser filen från uppgiften ovan, och skriver ut data om alla de igelkottar som har det namnet. Det ska använda funktionen **visa_igelkott**.

Uppgift 13 (4 p)

Skriv ett C-program som läser in tre rader med text, och sen skriver ut de två kortaste av dem. Om flera rader är lika långa, spelar det ingen roll vilken av dem som skrivs ut. Vi kommer aldrig att mata in längre rader än 107 tecken (plus radslutstecknet).

Uppgift 14 (4 p)



Kraftigt överviktig, enligt BMI-måttet.

BMI, "Body Mass Index", är ett mått som kan användas för att avgöra om en person har normal vikt. BMI definieras som vikten (i kilo) delat med kvadraten på längden (i meter). Det tar alltså inte hänsyn till kroppsbyggnad, utan bara vikt och längd.

Skriv ett C-program som upprepat läser in längd och vikt på en person, och skriver ut personens BMI.

Programmet ska avslutas när man matar in en person som har längden eller vikten noll.
