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

Tentamen i

Programmering grundkurs och Programmering C

för D1 m fl, även distanskursen

lördag 28 maj 2011

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/2010-2011-p2/, senast lördag 18 juni 2011.
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 uttryck?

a) 2 * 2 + 2 * 2

b) 2 + 2 * 2 + 2

c) 2 - 2 * 2 - 2

Uppgift 2 (3 p)

Skriv ett fullständigt C-program (med **#include** och allt) som läser in två flyttal, **x** och **y**, och sedan skriver ut de båda kvoterna **x**/**y** och **y**/**x**. Division med noll måste hanteras!

Ett körexempel, med användarens inmatning *kursiverad*:

```
Ange x: 2
Ange y: 0.5
x/y = 4.000000
y/x = 0.250000
```

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 3 (2 p)

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

```
a = 1; b = 2; x = 99.5; y = a;
while (x > a) {
    if (a == b || b == x) {
        a = a + 1;
    }
    else {
        x = x - 0.5;
    }
}
```

Uppgift 4 (2 p)

Skriv en funktion, **cirkelarea**, som tar en cirkels radie (**r**) som parameter, och som returnerar cirkelns area, som ges av formeln πr^2. (Konstanten **π** finns i include-filen **math.h** och heter **M_PI**.)

Uppgift 5 (2 p)

Skriv en **main**-funktion som läser in en radie, sen anropar funktionen **cirkelarea** för att räkna ut arean på en cirkel med den radien, och till sist skriver ut arean.

> 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 (3 p)

Skriv en funktion, **initialer**, som tar ett namn som parameter och som bestämmer namnets initialer. Namnet består alltid av minst ett ord, och om det innehåller flera ord är de åtskiljda av (enkla) blanktecken. Exempelvis har namnet **Kalle** initialen **K**, namnet **Kalle Anka** har initialerna **KA**, och namnet **Sven Karl-Oskar Svensson Bagge** har initialerna **SKSB**.

Funktionshuvud:

```
void initialer(char *helanamnet, char *initialer);
```

Uppgift 7 (2 p)

Skriv en **main**-funktion som läser in ett namn, sen anropar funktionen **initialer** för att få fram namnets initialer, och till sist skriver ut initialerna.

Uppgift 8 (1 p)

Ett äpple har en vikt i gram, till exempel **143.1**, och tillhör en viss äppelsort, till exempel **Granny Smith**.

Skapa datatypen **struct Apple**, som ska användas för att lagra data om ett äpple. Vi kan anta att ingen äppelsort har ett namn med mer än 20 tecken, inklusive mellanslag (som **Gyllenkroks astrakan**).

Uppgift 9 (1 p)

Definiera en variabel av typen **struct Apple** och initiera den med data om ett 200 gram tungt **Astrakan**-äpple.

Uppgift 10 (2 p)

Vi vill kunna visa äppelposternas innehåll på skärmen. Skriv en funktion som heter **visa_apple**, och som skriver ut data om ett äpple. Funktionen ska ta en äppelpost (eller, om du vill, en pekare till den) som parameter.

Uppgift 11 (2 p)

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

Du får själv välja vilket av dessa två funktionshuvuden du ska använda:

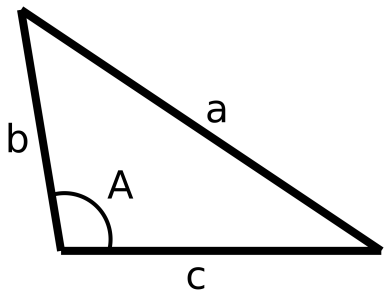
```
struct Apple las_apple()
void las_apple(struct Apple *p)
```

Uppgift 12 (2 p)

Skriv en main-funktion som har två lokala variabler av typen **struct Apple**, och som läser in data om två äpplen till dessa variabler med hjälp av funktionen **las_apple**. Därefter ska programmet skriva ut de två äpplenas data, i samma ordning som de matades in, med hjälp av funktionen **visa_apple**.

Uppgift 13 (5 p)

Det här är en triangel:



Om man vet längden på sidorna **b** och **c**, samt vinkeln **A**, kan man använda cosinussatsen för att beräkna längden på den motstående sidan **a**:

$$a = \sqrt{b^2 + c^2 - 2\,b\,c\cos(A)}$$

Skriv ett fullständigt C-program (med **#include** och allt) som upprepat läser in **b**, **c** och **A**, och sen skriver ut längden **a**. Programmet ska avslutas så fort man matar in en sida b som är mindre än eller lika med noll.

Ett körexempel, med användarens inmatning *kursiverad*:

```
Ange sidan b: 33
Ange sidan c: 42
Ange vinkeln A (i radianer): 1.74
a = 57.617686
Ange sidan b: 1000
Ange sidan c: 0.01
Ange vinkeln A (i radianer): 0.1
a = 999.990050
Ange sidan b: -13.7
```

Uppgift 14 (6 p)

En textfil innehåller flera miljarder decimaltal (vilket är mycket mer data än vad som får plats i datorns primärminne), med ett tal på varje rad:

```
3.14
2.71828
6.023
och så vidare
```

Skriv ett C-program som först frågar efter namnet på filen, och sedan läser filen och talar om dels hur många tal som totalt finns på filen, och dels hur många av talen som avviker med mer än 10 procent från talens medelvärde.

Om filen inte går att öppna, ska ett felmeddelande skrivas ut, och programmet ska avslutas.

Ledtråd: Man måste läsa igenom filen två gånger, först en gång för att räkna ut medelvärdet, och sen en gång för att räkna antalet avvikande tal.

Uppgift 15 (6 p)

Filen i uppgiften ovan innehåller alltså flera miljarder decimaltal, vilket är mycket mer data än vad som får plats i datorns primärminne. Men det visar sig att många av talen är upprepade, och det är inte mer än högst en miljon *olika* tal. En miljon tal får lätt plats i minnet.

Skriv ett C-program som först frågar efter namnet på filen, till exempel **alla-tal.txt**, sen frågar efter namnet på en andra fil, till exempel **unika-tal.txt**. Därefter ska programmet läsa talen från den första filen (**alla-tal.txt**), och skriva ut de unika talen (dvs med alla dubbletter borttagna) på den andra filen (**unika-tal.txt**).

(Vi har inga prestandakrav. Programmet får gärna ta lång tid att köra, bara det ger rätt utdata.)

Om filen inte går att öppna, ska ett felmeddelande skrivas ut, och programmet ska avslutas.

Ledtråd: Lagra de *unika* talen i minnet, till exempel i en array.
