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 26 februari 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 19 mars 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!

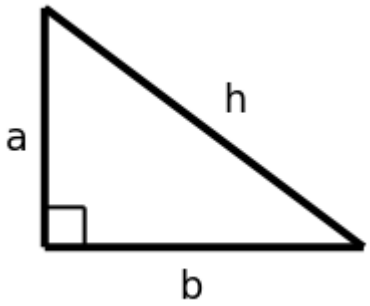
Prioritet och associativitet hos operatorerna i C

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 (3 p)

Det här är en rätvinklig triangel:



Om man vet längden på kateterna **a** och **b** i en rätvinklig triangel, kan man använda Pythagoras sats för att beräkna längden på hypotenusan **h**:

$$h = \sqrt{a^2 + b^2}$$

Skriv ett fullständigt C-program (med **#include** och allt) som läser in kateterna **a** och **b**, och sen skriver ut hypotenusan **h**.

Ett körexempel, med användarens inmatning *kursiverad*:

```
Ange a: 8
Ange b: 3.14
h = 8.594161
```

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 2 (2 p)

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

```
a = 1; b = a + 2; c = 3; d = 4;
while (a < b) {
    if (b == c && b == d) {
        a = a / 2 + 1 * 2;
        ++b;
    }
    else {
        a = a / 2 + 1 * 3;
        b--;
    }
}
```

Uppgift 3 (4 p)

Skriv ett fullständigt C-program (med **#include** och allt) som upprepat läser in ord, ända tills man matar in ordet **slut**, och för varje ord talar om hur många tecken det består av. När användaren skrivit ordet **slut**, ska programmet inte skriva ut något teckenantal, utan bara texten **Tack och hej.**

Ett körexempel, med användarens inmatning *kursiverad*:

```
Skriv ett ord: Bengt-Sofokles
14 tecken
Skriv ett ord: x
1 tecken
Skriv ett ord: slut
Tack och hej.
```

Man kan anta att inget ord är längre än 317 tecken.

Uppgift 4 (3 p)

a)

Skriv funktionen **medel**, som tar två heltal som argument, och returnerar medelvärdet av dem, men som ett flyttal. (Medelvärdet av heltalen **1** och **2** ska alltså bli **1.5**.)

b)

Skriv en **main**-funktion som läser in två heltal, sen anropar funktionen **medel** för att få reda på deras medelvärde, och till sist skriver ut det medelvärdet.

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 5 (6 p)

a)

Skriv funktionen **mellan**, som tar fyra argument: två flyttal (vi kan kalla dem **min** och **max**), en array av flyttal, och ett heltal som anger hur många flyttal som finns i den arrayen. Funktionen ska returnera ett heltal som anger hur många av talen i arrayen som ligger mellan de två talen **min** och **max**.

b)

Skriv en **main**-funktion som först läser in tusen flyttal till en array, sen anropar funktionen **mellan** för att få reda på hur många av dessa tal som ligger mellan **3** och **3.5**, och till sist skriver ut det antalet.

Uppgift 6 (1 p)

När man mäter något, till exempel bredden på ett bord, kan man sällan mäta exakt, utan man har alltid ett visst fel, som man ibland kan ange gränser för. Vi kanske mäter att bordet är **99.2** cm brett, men vi kan egentligen bara vara säkra på att bredden ligger mellan **99.15** och **99.25** cm.

Skapa datatypen **struct Tal**, som ska användas för att representera ett tal med felgränser. Denna post ska innehålla tre flyttal: det uppskattade värdet, och sen min- och max-gränsen för talets verkliga värde.

Uppgift 7 (1 p)

Definiera en variabel av typen **struct Tal** och initiera den med det uppskattade värdet **100.0**, min-gränsen **82.3** och max-gränsen **144.3**. (Ja, vi hade en ganska dålig linjal.)

Uppgift 8 (2 p)

Vi vill kunna visa talposternas innehåll på skärmen. Skriv en funktion som heter **visa_tal**, och som skriver ut data om ett tal. Funktionen ska ta en talpost (eller, om du vill, en pekare till den) som parameter. Här är ett exempel på hur utskriften kan se ut:

```
100.000000 (min: 82.300000, max: 144.300000)
```

Uppgift 9 (2 p)

Skriv en funktion som heter **las_tal**, och som läser in ett tal, med felgränser. 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 Tal las_tal()
void las_tal(struct Tal *p)
```

Uppgift 10 (2 p)

När man adderar två tal med felgränser, ska man också beräkna ett min- och ett maxvärde för resultatet. Min-gränsen är summan av de två talens min-gränser, och max-gränsen är summan av de två talens max-gränser. Skriv en funktion **plus**, som tar två talposter som argument, och som returnerar en tal-post med talens summa, med korrekta felgränser.

Uppgift 11 (2 p)

Skriv en main-funktion som har två lokala variabler av typen **struct Tal**, och som läser in data om två tal till dessa variabler med hjälp av funktionen **las_tal**. Därefter ska programmet beräkna summan med hjälp av funktionen **plus**, och skriva ut den med funktionen **visa_tal**.

Uppgift 12 (3 p)

Vi ska skriva funktionen **storre**, som jämför två tal för att se om det första är större än det andra.

När man jämför två tal som har felgränser kan det bli lite komplicerat. Om man inte vet säkert hur stora talen är, kan man inte alltid veta vilket som är störst.

Exempelvis vet vi att ett tal med min-gränsen **3** och max-gränsen **4** är mindre än ett tal med min-gränsen **5** och max-gränsen **6**. Men om vi jämför ett tal med min-gränsen **3** och max-gränsen **5**, och ett tal med min-gränsen **4** och max-gränsen **6**, kan vi inte veta vilket som egentligen är störst.

Skriv funktionen **storre**, som tar två tal-poster som argument, och som returnerar **1** om det första talet är större än det andra, **0** om det första talet är mindre än eller lika med det andra, och **-1** om det inte går att avgöra.

Uppgift 13 (4 p)

Skriv ett C-program som läser in tal (med felgränser) med hjälp av funktionen **las_tal**, och sparar allihopa på en fil. Välj själv hur inmatningen ska avslutas. (Det ska inte vara ett förutbestämt antal tal.)

Uppgift 14 (5 p)

Skriv ett C-program som först läser in ett tal med funktionen **las_tal**. Det talet ska sen användas som gräns. Därefter ska programmet läsa filen från uppgiften ovan, och skriva ut tre saker: hur många av talen som *säkert* är större än gränsen, hur många som *kan* vara större än gränsen, och hur många som garanterat *inte* är större än gränsen. Använd funktionen **storre**.
