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

torsdag 19 augusti 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 torsdag 9 september 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) 4 + 3 * 2 - 1

b) 4 - 3 - 2 - 1

c) 4 / 3 + 2 % 1

Uppgift 2 (1 p)

Variablerna **x**, **y**, **z** och **t** är av typen **double**. Vilka värden har variablerna efter att följande kod har körts?

```
x = 3.5; y = 2.0; z = 2; t = x + y;
if (x > y)
    while (y == z)
        if (t > x)
            t = t - y;
        else
            y -= y;
```

Uppgift 3 (2 p)

Skriv en funktion **median**, som tar tre flyttal som argument och returnerar det mellersta storleksmässigt. Vi kan anta att alla tre talen är olika.

Uppgift 4 (2 p)

Skriv en **main**-funktion som först läser in tre flyttal från användaren, sedan anropar funkionen **median** för att få fram mellantalet, och till sist skriver ut detta tal.

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.

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 5 (4 p)

Vi ska göra ett C-program som hanterar anslagstavlor.

Först ska vi skapa datatypen **struct Anslag**, som används för att representera anslag. Ett anslag har en rubrik, som är en enda rad och innehåller högst 60 tecken, och en text, som kan bestå av högst 400 tecken. Texten kan innehålla radslutstecken.

Sen ska vi skapa datatypen **struct Anslagstavla**, som används för att representera anslagstavlor. En anslagstavla innehåller anslag, upp till maximalt 60 stycken, och de ska lagras i anslagstavleposten som en array av **struct Anslag**.

Vi ska också använda **#define** för att skapa makron för max-rubriklängden, max-textlängen och maximala antalet anslag per tavla.

Skapa de båda posttyperna och de tre makrona, i rätt ordning så att det går att kompilera.

Uppgift 6 (2 p)

Definiera en variabel av typen **struct Anslag** och initiera den med data om ett anslag med rubriken **Bok säljes** och den här texten:

> **Boken "Vägen till C" av Bilting & Skansholm.**
> **Väl använd men i gott skick.**

Uppgift 7 (3 p)

Vi vill kunna visa anslagsposternas innehåll på skärmen. Skriv en funktion som heter **visa_anslag**, som skriver ut data om ett anslag. Funktionen ska ta en anslagspost (eller, om du vill, en pekare till den) som parameter. Här är ett exempel på hur utskriften ska se ut:

```
Bok säljes
==========
Boken "Vägen till C" av Bilting & Skansholm.
Väl använd men i gott skick.
```

För full poäng på uppgiften ska rubriken strykas under med precis lagom många likhetstecken, som i exemplet.

Uppgift 8 (4 p)

Skriv en funktion som heter **las_anslag**, och som läser in data om ett anslag. 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 Anslag las_anslag()
```

eller så här:

```
void las_anslag(struct Anslag *p)
```

Eftersom texten på anslaget kan bestå av flera rader, behövs något sätt att avsluta inmatningen av en text. Ett sätt kan vara att mata in en tom rad.

Uppgift 9 (2 p)

Skriv en main-funktion som har tre lokala variabler av typen **struct Anslag**, och som läser in data om tre anslag till dessa variabler med hjälp av funktionen **las_anslag**. Därefter ska programmet skriva ut de tre anslagen med hjälp av funktionen **visa_anslag**, men i omvänd ordning (dvs med det sist inmatade anslaget först).

Uppgift 10 (3 p)

Vi vill lagra data om ett anslag på en fil. Skriv funktionen **spara_anslag**, som sparar ett anslag på en fil. Funktionen ska ta en anslagspost (eller, om du vill, en pekare till den) som parameter, samt namnet på den fil där anslaget ska sparas. Välj själv om det ska vara en textfil eller en binärfil. Om filen inte kan öppnas, ska ett felmeddelande skrivas ut och programmet avslutas.

Uppgift 11 (3 p)

Skriv en funktion som heter **hamta_anslag** ("hämta anslag"), som läser in data om ett anslag från en fil av den typ som skrevs i uppgiften ovan. Funktionen ska ta lämpliga parametrar, men en av parametrarna ska vara namnet på den fil som ska läsas. Om filen inte kan öppnas, ska ett felmeddelande skrivas ut och programmet avslutas.

Uppgift 12 (8 p)

Textfilen **filnamn.txt** innehåller ett stort antal rader, där varje rad består av namnet på en fil där ett anslag lagrats med funktionen **spara_anslag**. Inget filnamn är längre än 347 tecken. Skriv ett C-program som (med funktionen **visa_anslag**) visar det anslag som har den längsta texten. (Rubrikens längd räknas inte.) Om det finns flera anslag med samma (längsta) textlängd, spelar det ingen roll vilket av dem vi skriver ut. Om någon av de inblandade filerna inte kan öppnas, ska ett felmeddelande skrivas ut och programmet avslutas.

Uppgift 13 (5 p)

Skriv ett program som först läser in två positiva heltal, och sedan skriver ut alla tal från och med det minsta inlästa talet till och med det största inlästa talet, och till sist skriver ut summan av alla *udda* respektive *jämna* utskrivna tal.

Ett körexempel, med användarens inmatning understruken:

```
Ge ett heltal: 5
Ge ett heltal till: 2
2, 3, 4, 5
Summan av udda tal: 8
Summan av jämna tal: 6
```
