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

Tentamen i
Programmering C

(distanskurs)

onsdag 7 juni 2006 kl 8:00 - 13:00 i L001

Hjälpmedel:	Inga hjälpmedel.
Poängkrav:	Maximal poäng är 40. För godkänt betyg (3 respektive G) krävs 20 poäng.
Resultat och lösningar:	Meddelas på kursens hemsida senast måndag 19 juni 2006.
Visning och frågestund:	Måndag 19 juni 2006 kl 12:00-12:30 i mitt rum (T2220). Efter visningen kan tentorna hämtas på expeditionen.
Examinator och jourhavande:	Thomas Padron-McCarthy, telefon 0707-347013.

- Skriv tydligt och klart. Lösningar som inte går att läsa kan naturligtvis inte ge några poäng. Oklara 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. Gjorda antaganden får inte förändra den givna uppgiften.

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)

Skriv formeln nedan som en tilldelningssats i C.

$$z = x + \frac{y}{\sqrt{x^2 + z}}$$

Uppgift 2 (1 p)

Vilka värden har följande uttryck?

a) 1 + 2 / 3

b) 1 + 2.0 / 3

c) 1 + (float)2 / 3

d) 1 + (float)(2 / 3)

Uppgift 3 (1 p)

i är en heltalsvariabel. Ange värdet på i då följande kod körts.

```
i = 1;
do {
    printf("Hej!\n");
    i = i * 2;
} while (i < 6);
```

Uppgift 4 (1 p)

Biblioteket i Grythyttebymobruk behöver hålla reda på sina böcker. De flesta skulle nog skaffat en särskild databashanterare, eller kanske ett kortregister, men Grythyttebymobruks bibliotek bestämmer sig för att använda ett C-program.

Programmet behöver först och främst en posttyp (med C-terminologi en **struct**) som kan användas för att lagra data om en bok. Det som ska finnas med i posten är:

- Författarens förnamn, som kan vara upp till tio tecken långt.
- Författarens efternamn, som kan vara upp till tjugo tecken långt.
- Bokens titel, som kan vara upp till hundra tecken lång.
- Bokens vikt, som anges med ett flyttal. Vikten mäts i kilo.
- Bokens nummer, som är ett heltal. Varje bok har ett unikt nummer, så två böcker kan inte ha samma nummer.

Definiera posttypen. Den ska heta **struct Bok**.

Uppgift 5 (1 p)

Definiera en variabel av typen **struct Bok** och initiera den med data om bibliotekets bok nummer **138**, som heter **I nationens intresse** och som är skriven av **Jan Guillou**. Den väger 0.3 kilo.

Uppgift 6 (2 p)

Skriv en funktion som heter **visa_bok**, som skriver ut en bok (som i uppgift 4) på skärmen. Funktionen tar bok posten som parameter, och skriver ut den. Exempel på hur urskriften ska se ut:

```
Nummer: 138
Författare: Guillou, Jan
Titel: I nationens intresse
Vikt: 0.3 kg
```

Så här ser funktionshuvudet ut:

```
void visa_bok(struct Bok boken)
```

Uppgift 7 (3 p)

Böcker måste ofta sorteras, och i biblioteket ställer man böckerna i bokstavsordning efter författarens efternamn. Om två böcker har samma författar-efternamn, sorterar man dem i bokstavsordning efter förnamnen. Om även förnamnen är lika, sorterar man dem i bokstavsordning efter böckernas titlar. Om även titlarna skulle vara lika, antar vi att det är två exemplar av samma bok, och då spelar det ingen roll i vilken ordning de står.

Skriv en funktion som heter **i_ordning**, som tar två bokposter som argument. Så här ser funktionshuvudet ut:

```
int i_ordning(struct Bok bok1, struct Bok bok2)
```

Om böckerna står i rätt ordning, dvs att bok1 enligt reglerna ska stå före bok2 i bokhyllan, eller om det inte spelar någon roll i vilken ordning de två böckerna ska stå, ska funktionen returnera ett sant värde.
Om böckerna står i fel ordning, ska funktionen returnera ett falskt värde.

Man kan använda funktionen **strcmp** för att jämföra två strängar. Den tar två strängar som argument, och returnerar ett heltal som är:

- mindre än noll om den första strängen ska stå före den andra vid alfabetisk sortering
- noll om strängarna är lika
- större än noll om den första strängen ska stå efter den andra vid alfabetisk sortering

Uppgift 8 (2 p)

Skriv en funktion som heter **las_bok**, och som läser in data om en bok (enligt uppgift 4 ovan). 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 Bok las_bok()
```

eller så här:

```
void las_bok(struct Bok* p)
```

Uppgift 9 (1 p)

Talet **12345678** skulle kunna utläsas **12 miljoner och lite till**. Skriv en funktion som heter **hela_miljoner**, som tar ett positivt heltal som argument och som returnerar hur många hela miljoner det motsvarar. Exempel:

- hela_miljoner(12345678) = 12
- hela_miljoner(12999999) = 12
- hela_miljoner(144000) = 0

Uppgift 10 (2 p)

Talet **98765432** skulle kunna utläsas **98 miljoner och lite till**, men om man avrundar till närmaste hela miljon blir det **99 miljoner**. Skriv en funktion som heter **cirka_miljoner**, som tar ett positivt heltal som argument och som avrundar till närmast hela antal miljoner. Exempel:

- cirka_miljoner(12345678) = 12
- cirka_miljoner(12999999) = 13
- cirka_miljoner(144000) = 0

Tal som slutar på 500000, och alltså ligger mitt emellan två miljoner, ska avrundas till ett jämnt antal miljoner (alltså ett antal som är jämnt delbart med två). Exempel:

- cirka_miljoner(11500000) = 12
- cirka_miljoner(12500000) = 12
- cirka_miljoner(13500000) = 14

Uppgift 11 (5 p)

Skriv ett C-program som frågar efter användarens årsinkomst och förmögenhet. Båda talen är heltal. Detta ska upprepas tills användaren matar in årsinkomsten 0.

Efter varje inmatning av en årsinkomst och en förmögenhet ska programmet tala om ungefär hur många miljoner användaren har i årsinkomst och förmögenhet. Använd funktionen **cirka_miljoner**.

I den här och alla andra uppgifter på tentan gäller: Om du ska anropa en funktion från en tidigare uppgift, behöver du inte skriva koden för den funktionen på nytt. Du får också anropa funktionen även om du inte gjort uppgiften där man skulle skriva den.

Ett exempel (med användarens inmatning i *kursiv stil*):

```
Ange årsinkomst: 2600000
Ange förmögenhet: 7500000
Du tjänar ungefär 3 miljoner om året,
och äger ungefär 8 miljoner.
```

De flesta av oss får förstås betydligt tråkigare siffror. Exempel:

```
Ange årsinkomst: 260000
Ange förmögenhet: 80000
Du tjänar ungefär 0 miljoner om året,
och äger ungefär 0 miljoner.
```

Det gör inget om programmet ibland skriver "1 miljoner".

Uppgift 12 (4 p)

Jag håller på och renoverar hemma, och då ingår ommålning av väggarna, läggning av nytt golv, samt byte av golvlister. För varje rum behöver jag räkna ut hur mycket väggfärg, hur mycket golvmatta och hur många meter golvlist jag ska köpa.

Skriv ett C-program som hjälper till med detta! Det ska fråga efter rummets bredd, längd och höjd i meter (tre flyttal), och sen tala om hur många liter väggfärg som behövs, samt hur många kvadratmeter golvmatta och hur många meter golvlist.

Ledning:

- Räkna med att det går ett en liter färg per kvadratmeter vägg som ska målas. Vi tänker oss att alla rummen är fyrkantiga och regelbundna, och att vi inte bryr oss om att dra bort ytan för fönster och dörrar. Mängden väggfärg som behövs fås då genom att ta bredden gånger höjden gånger två, plus längden gånger höjden gånger två.
- Mängden golvmatta är lika med golvytan, som fås genom att ta bredden gånger längden.
- Längden på golvlisterna är lika med två gånger längden plus två gånger bredden.

Exempel på dialog med programmet:

```
Ange rummets bredd: 2.10
Ange rummets längd: 3
Ange rummets höjd: 2.40

Liter målarfärg: 24.48
Kvadratmeter golvmatta: 6.30
Meter golvlister: 10.20
```

Uppgift 13 (4 p)

Biblioteket i Grythyttebymobruk ska mata in alla sina böcker. Skriv ett C-program som låter användaren mata in böcker och som sparar dem på en binärfil.

Programmet ska läsa en bok i taget med hjälp av funktionen **las_bok** från uppgift 8. Allteftersom böckerna läses in, ska de sparas på binärfilen. På binärfilen lagras poster av typen **struct Bok**. Binärfilen ska heta **bocker.bin**.

Välj själv ett lämpligt sätt för hur inmatningen ska avslutas, så att användaren på något vis kan tala om för programmet att nu har hon inga fler böcker att mata in.

Uppgift 14 (5 p)

Inmatningsprogrammet i uppgift 11 sorterar inte böckerna, utan de hamnar på filen i den ordning de matades in. Skriv ett C-program som sorterar böckerna i bokstavsordning. (Se uppgift 7 för närmare detaljer om vad som här menas med "bokstavsordning".)

Gör så här:

1. Läs in alla bokposterna från filen **bocker.bin** till en array.
2. Sortera arrayen. Det är tillåtet att använda funktionen **i_ordning**.
3. Spara alla bokposterna på en ny binärfil, **sorterat.bin**.

Du kan anta att biblioteket kan ha 1000 böcker, men inte fler.

Uppgift 15 (7 p)

Det vanliga medelvärdet av ett antal tal räknas ut genom att man adderar talen och sen delar summan med antalet. Medelvärdet av talen 81, 83, 85 och 82 är exempelvis 82.75.

För en serie av tal kan man räkna ut ett **glidande medelvärde**. Det glidande medelvärdet är medelvärdet av de sista talen i serien, till exempel de tio sista. Alltefter som man lägger till tal i slutet av serien, blir det alltså nya tal som används för att beräkna det glidande medelvärdet.

Ett glidande medelvärde kan man ha nytta av när man väger sig. Om man väger sig varje dag märker man att vikten kan variera med flera kilo från dag till dag. Det beror inte på att man ändrat mängden muskler eller fett i kroppen, utan bara på vattenbalansen, dvs hur mycket vatten man råkar ha i kroppen just då. Om man håller diet för att gå ner i vikt, kan de stora variationerna från dag till dag vara besvärliga. Har jag verkligen gått upp två kilo sen igår, trots att jag håller diet?

Då är det bättre att använda ett glidande medelvärde över till exempel 30 dagar. Man väger sig fortfarande varje dag, men den vikt man tänker sig att man har är inte resultatet av den senaste vägningen, utan medelvärdet av de senaste trettio vägningarna. Då blir variationerna i vikten mycket mindre, och viktkurvan jämnare.

Vi antar att vi väger oss varje dag, och skriver in vikten på en egen rad i en textfil. Textfilen heter **vikter.txt**. Skriv ett C-program som läser den textfilen och beräknar medelvärdet av de trettio sista vikterna i filen.

Förslag på lösning:

- Låt programmet innehålla en array med plats för trettio flyttal, samt ett heltal som anger hur många flyttal som just nu faktiskt finns i arrayen.
- Öppna och läs filen, och lägg in varje inläst tal i arrayen.
- Om arrayen är full, så släng först bort det äldsta talet, för att skapa plats för det nya inlästa talet.
- När alla talen lästs från filen, beräknar vi medelvärdet av talen i arrayen, och skriver ut det.

Om filen innehåller mindre än 30 tal, ska programmet beräkna medelvärdet av de tal som fanns.

För full poäng på uppgiften ska programmet kunna upptäcka följande fel, och rapportera dem med en rättvisande och informativ felutskrift:

- Filen **vikter.txt** gick inte att öppna. (Märks genom att **fopen** returnerar **NULL**.)
- Filen var tom. (Medelvärdet av noll stycken tal är ju inte definierat.)
- Filen innehöll något som inte var ett giltigt flyttal. (Tänk på att **fscanf** returnerar antalet **%**-saker som den lyckades läsa, eller **EOF** om filen var slut.)