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 23 maj 2009 kl 8:15 - 13:15

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.<br>För godkänt betyg (3 respektive G) krävs 20 poäng. |
| **Resultat och lösningar:** | Meddelas via e-post eller på campuskursens hemsida, http://www.aass.oru.se/~tpy/c/2008-2009-p2/, senast lördag 13 juni 2009. |
| **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 (3 p)

Skriv ett komplett C-program (med **#include** och allt) som först frågar efter ett heltal, och sedan läser in så många reella tal som angavs av heltalet. Avslutningsvis ska de reella talens summa skrivas ut.

Exempel:

```
Antal reella tal att mata in: 3
Mata in 3 reella tal:
1.19
7
2.19
Summa: 10.38
```

> I den här och alla andra uppgifter på tentan, antar vi för enkelhets skull att användaren aldrig matar in några felaktiga värden (till exempel att det ska vara **-7** tal, eller det reella talet **Kalle**).

# Uppgift 2 (3 p)

Det här är en vattenkokare, som man till exempel kan värma tevatten i:


Skriv ett C-program som upprepat läser in volym och effekt för vattenkokare, ända tills man skriver in en vattenkokare vars volym eller effekt är noll. Som avslutning ska programmet skriva ut effekten för den vattenkokare som har störst voym.

# Uppgift 3 (3 p)

Skriv en funktion som heter **array_count** och som tar tre argument: en osorterad array ("vektor") med flyttal av typen **double**, ett heltal som anger antalet tal i arrayen, och ett flyttal som ska räknas. Funktionen ska returnera hur många av talen i arrayen som var lika med det medskickade flyttalet.

# Uppgift 4 (2 p)

Skriv en **main**-funktion som anropar funktionen **array_count** från uppgiften ovan, och skriver ut resultatet.

**main**-funktionen måste skicka med lämpliga argument i anropet, så att den anropade funktionen kan returnera ett meningsfullt svar. Välj själv var värdena som skickas med ska komma ifrån: inmatning från användaren, konstanter, eller kanske något annat.

# Uppgift 5 (3 p)

Vi ska simulera en termostat. Skriv ett C-program som först frågar efter en önskad temperatur, och sen upprepat frågar efter aktuell temperatur. Om det är kallare än önskat (dvs om den aktuella temperaturen är lägre än den önskade temperaturen) ska termostaten slå på elementet, och om det är varmare än önskat ska den slå på kylningen. Mängden tillförd värme och kyla ska styras av temperaturskillnaden enligt följande tabell:

| Temperaturskillnad | Resultat |
|---|---|
| Mindre än 1 grads skillnad | Ingenting |
| Högst 3 grader för kallt | Elementet på lite |
| Högst 3 grader för varmt | Lite kylning |
| Mer än 3 grader för kallt | Elementet på för fullt |
| Mer än 3 grader för varmt | Kylning för fullt |

Körexempel:

```
Ange önskad temperatur: 18
Ange aktuell temperatur: 18.1
Ingen åtgärd.
Ange aktuell temperatur: 18.99
Ingen åtgärd.
Ange aktuell temperatur: 19
Lite kylning.
Ange aktuell temperatur: 21
Lite kylning.
Ange aktuell temperatur: 21.2
Kylning för fullt!
Ange aktuell temperatur: 0
Elementet på för fullt!
Ange aktuell temperatur: -3
Elementet på för fullt!
...
```

# Uppgift 6 (5 p)

Skriv ett C-program som upprepat läser in tre hela tal och skriver ut det minsta av dessa. Programmet ska avslutas direkt utan att fråga efter de två övriga talen då 0 inmatas som första tal.

Körexempel:

```
Ge första talet (avsluta med 0): 23
Ge andra talet: 18
Ge tredje talet: 20
Minsta talet = 18
Ge första talet (avsluta med 0): 13
Ge andra talet: 28
Ge tredje talet: 20
Minsta talet = 13
Ge första talet (avsluta med 0): 0
```

# Uppgift 7 (5 p)

Skriv ett C-program som läser in rader, ända tills användaren matar in en tom rad. När inmatningen är slut ska programmet skriva ut de av raderna som innehåller exakt tre förekomster av bokstaven (stora) **T**.

Vi kan anta att ingen rad i inmatningen kommer att vara längre än 719 tecken (plus radslutstecken), och att man aldrig kommer att mata in fler än 618 rader.

# Uppgift 8 (16 p)

YouTube är en webbplats där man kan titta på, och lägga upp, videosnuttar. För att veta vilka av videosnuttarna som är populärast, håller YouTube reda på hur många gånger användarna klickat på, och alltså tittat på, varje videosnutt.

YouTube sparar data om klickningarna i en fil som heter **indata.txt** och som ser ut så här:

```
226 1
13 10
226 1
13 1
17290219 2
13 37
```

Varje rad innehåller ett videonummer och ett antal klick:

- Video nummer 226 har fått ett klick.
- Video nummer 13 har fått 10 klick.
- Video nummer 226 har fått ytterligare ett klick.
- Video nummer 13 har fått ytterligare ett klick.
- Video nummer 17290219 har fått två klick.
- Video nummer 13 har fått ytterligare 37 klick.

Vi vill sammanställa uppgifterna till en ny fil **utdata.txt**, som har samma format men där vi slagit ihop antalet klick per video, så att varje videonummer bara finns på en rad:

```
226 2
13 48
17290219 2
```

Några ytterligare saker:

- Ordningen mellan de olika videonumren i den nya filen spelar ingen roll.
- Videonumren kan vara stora tal, men får plats i en heltalsvariabel av typen **int**.
- Antalet klick per video kan vara stort, men får plats i en heltalsvariabel av typen **int**.
- Det finns högst en miljon videosnuttar på YouTube.
- Vi har inga särskilda krav på snabbhet.

| |
|---|
| Följande deluppgifter ger en steg-för-steg-beskrivning av hur programmet kan skrivas. Om du vill kan du i stället göra en helt egen lösning, vilket också ger full poäng, om programmet fungerar och är begripligt. |

a) (1p)

Skapa en posttyp som heter **struct VideoKlick** och som innehåller två heltal: ett videonummer och ett antal klick. Den ska användas för att lagra antalet klick för en viss video.

b) (1p)

Använd **#define** för att definiera konstanten ("makrot") **MAX_ANTAL_VIDEOSAR**, som ska ha värdet en miljon.

c) (1p)

Skapa en array ("fält") som heter **klick**, som innehåller **MAX_ANTAL_VIDEOSAR** stycken poster av typen **struct VideoKlick**. Skapa också en heltalsvariabel som heter **antal_videosar**, som får startvärdet noll, och som anger antalet poster som vi lagrat i den arrayen.

d) (1p)

Men behöver vi verkligen variabeln **antal_videosar**? Vi har ju redan angett en storlek på arrayen? Förklara!

e) (2p)

Skriv funktionen **hitta_position**, som tar ett videonummer som argument, och som söker igenom arrayen **klick** och returnerar den position i arrayen där posten för det videonumret finns. Den returnerade positionen är ett heltal. Om videonumret inte hittades, ska funktionen returnera **-1**.

f) (3p)

Skriv funktionen **ny_video**, som tar ett videonummer som argument, och som lägger in en ny post med det videonumret i arrayen **klick**. Antalet klick i den posten ska sättas till noll. Funktionen ska returnera positionen i arrayen där den nya posten placerades.

Om arrayen redan är full, ska ett felmeddelande skrivas ut, och programmet ska

avslutas. (Tips: Anropa funktionen **exit**.)

g) (2p)

Skriv funktionen **addera_klick**, som tar två argument: ett videonummer och ett antal klick. (Det motsvarar en rad i indatafilen.) Funktionen ska anropa **hitta_position** för att söka rätt på det angivna videonumret i arrayen **klick**. Om videonumret inte hittades, ska den anropa **ny_video** för att lägga in en ny post för videonumret. Avslutningsvis ska antalet klick för videonumret ökas med det medskickade antalet klick.

h) (4p)

Och så behövs det en **main**-funktion också. Den ska öppna infilen, läsa den, och anropa **addera_klick** för varje inläst rad. Därefter ska den öppna utfilen, och skriva den.

Om någon fil inte gick att öppna, ska ett felmeddelande skrivas ut, och programmet ska avslutas.

i) (1p)

Vilka **#include**-rader behövs i början av programmet?

---