Örebro universitet
Institutionen 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 17 januari 2015

Gäller som tentamen för:
DT1029 Datateknik A, Programmering grundkurs, provkod 0100
DT1030 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. För den som följt campuskursen hösten 2014 ger varje i tid inlämnad inlämningsuppgift med deadline en extra poäng. Den som *inte* gått campuskursen hösten 2014 får dessa (tre) extrapoäng ändå.
Resultat och lösningar:	Meddelas via e-post senast lördag 7 februari 2015.
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

Några användbara biblioteksfunktioner

stdlib.h

```
int rand(void);
void srand(unsigned int seed);
void *malloc(size_t size);
void *realloc(void *ptr, size_t size);
void free(void *ptr);
void exit(int status);
void qsort(void *base, size_t nmemb, size_t size,
           int(*compar)(const void *, const void *));
```

stdio.h

```
FILE *fopen(const char *path, const char *mode);
int fclose(FILE *stream);
int getc(FILE *stream);
int getchar(void);
int ungetc(int c, FILE *stream);
char *fgets(char *s, int size, FILE *stream);
char *gets(char *s);
int putc(int c, FILE *stream);
int printf(const char *format, ...);
int fprintf(FILE *stream, const char *format, ...);
int sprintf(char *str, const char *format, ...);
int snprintf(char *str, size_t size, const char *format, ...);
int scanf(const char *format, ...);
int fscanf(FILE *stream, const char *format, ...);
int sscanf(const char *str, const char *format, ...);
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);
```

string.h

```
size_t strlen(const char *s);
char *strcpy(char *dest, const char *src);
char *strncpy(char *dest, const char *src, size_t n);
int strcmp(const char *s1, const char *s2);
int strncmp(const char *s1, const char *s2, size_t n);
char *strcat(char *dest, const char *src);
char *strncat(char *dest, const char *src, size_t n);
char *strstr(const char *haystack, const char *needle);
void *memmove(void *dest, const void *src, size_t n);
```

ctype.h

```
int isalnum(int c);
int isalpha(int c);
int isblank(int c);
int isdigit(int c);
int islower(int c);
int isprint(int c);
int ispunct(int c);
int isspace(int c);
int isupper(int c);
```

math.h

```
double sqrt(double x);
double pow(double x, double y);
```

Uppgift 1 (1 p)

Vilka värden har följande C-uttryck?

a) `1 + 2 + 3 + 4`

b) `1 * (2 * 3 + 4)`

c) `(1 * 2) * 3 + 4`

d) `1 + 2 * 3 + 4`

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 = 2;
c = a + b;
d = 2 * 3+4;
while (a < b) {
    a = a + 1;
    b = b - 2;
    c = c;
}
printf("d = 7");
```

Uppgift 3 (5 p)

Modulo-operatorn **%** ger heltalsresten vid division. Man kan använda den för att avgöra om ett tal är jämnt delbart med ett annat tal.

Skriv ett komplett C-program (med **#include** och allt) som skriver ut talen från 1 till 100 (inklusive 1 och 100), men för tal som är jämnt delbara med tre ska programmet i stället skriva "Fizz", och för tal som är jämnt delbara med fem ska det skriva "Buzz". För tal som är jämnt delbara med både tre och fem ska programmet skriva "FizzBuzz".

Programmet måste bygga på en loop, så en lösning som exempelvis består av hundra printf ger noll poäng.

I den här och alla andra uppgifter på tentan gäller: Man kan strunta i detaljer som bara behövs just när man utvecklar konsolprogram i Visual Studio, som konstiga teckenkoder för ÅÄÖ, eller att fönstret med programkörningen försvinner när programmet avslutas.

Uppgift 4 (4 p)

Vi vill kunna beräkna följande uttryck:

$$\frac{3}{4}\sqrt{\frac{(x-y)}{(x+y)^2}-1}$$

Skriv därför ett komplett C-program (med **#include** och allt) som först läser in värden på x och y, beräknar uttrycket, och till sist skriver ut uttryckets värde. Beräkningarna ska ske med flyttal.

Om deluttrycket **x+y** är noll, eller om hela deluttrycket under roten är mindre än noll, går uttrycket inte att beräkna. I så fall ska programmet inte försöka beräkna uttrycket, utan det ska i stället skriva ut ett informativt och rättvisande felmeddelande om saken.

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 (5 p)

a) Skriv en funktion som tar två argument: en array av flyttal, och ett heltal som anger antalet tal i arrayen. Funktionen ska returnera summan (ett flyttal) av det största och det minsta talet i arrayen.

b) Skriv en main-funktion som låter användaren mata in tio flyttal, som lagras i en array. Därefter ska main-funktionen anropa funktionen från a-uppgiften, och skriva ut resultatet.

I den här och alla andra uppgifter på tentan gäller: Om du behöver använda något från en tidigare uppgift eller deluppgift, till exempel utnyttja en datatyp eller 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 (5 p)

a) Skriv en funktion som tar två strängar som argument, och som returnerar ett sant värde om den andra strängen är lika med den första strängen, men vänd baklänges. Annars ska funktionen returnera ett falskt värde. Exempelvis ska argumenten **"Hej!"** och **"!jeH"** returnera sant.

b) Skriv en main-funktion som låter användaren mata in två strängar. Därefter ska main-funktionen anropa funktionen från a-uppgiften, och skriva ut resultatet. Man kan räkna med att ingen sträng är längre än 100 tecken.

Uppgift 7 (8 p)

a) Skriv definitionen av en post (med C-terminologi en **struct**) som vi kan använda för att lagra tätorter med namn, folkmängd och position. Posten ska innehålla ett namn i form av en sträng, folkmängden som ett heltal, och latitud och longitud i form av flyttal.

b) Skapa en variabel som är av posttypen ovan och som innehåller data om tätorten **Stora Mellösa** med **776** invånare, latutiden **59.21** och longituden **15.50**.

c) Skriv en funktion som tar två argument: en array av tätortsposter, och ett heltal som anger antalet tal i arrayen. Funktionen ska returnera den genomsnittliga folkmängden (ett flyttal) för tätorterna.

d) Vi vill testa funktionen från uppgiften ovan. Skriv därför en **main**-funktion, som anropar den med lämpliga argument, och skriver ut resultatet. Vi ska bara anropa funktionen en gång, så det finns alltså bara ett enda testfall per funktion. Försök göra detta testfall så bra som möjligt!

Uppgift 8 (10 p)

Ibland får man rådet att inte skriva ner lösenord, utan i stället memorera dem. Det var kanske ett bra råd för trettio år sen, när en vanlig datoranvändare hade ett eller ett par lösenord att komma ihåg, men idag har man kanske konton på ett hundratal olika webbplatser, och man bör ju ha olika lösenord på allihop. Därför behöver man ha något sätt att hålla reda på lösenorden, och vi ska göra ett C-program för att hålla reda på dem. Vi sparar dem på en fil.

a) Skriv ett program som vi kan använda för att spara ett lösenord. Användaren ska mata in två saker: namnet på webbplatsen eller datorn som lösenordet gäller, och själva lösenordet. Programmet ska spara namnet och lösenordet i filen. Vi kan anta att både namn och lösenord består av ett enda ord, dvs utan mellanslag eller radbyten. Vi kan också anta att inget namn eller lösenord är längre än 100 tecken.

Dessutom kan vi anta att användaren aldrig ändrar ett lösenord, så vi behöver inte hantera problemet att det redan finns ett lösenord lagrat för en viss webbplats eller dator.

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

Tips: **mode**-strängen **"a"** till funktionen **fopen** öppnar en fil för att skriva till den, på samma sätt som **"w"**, men **"a"** lägger till nya data i slutet av filen utan att skriva över det som redan finns på filen.

b) Skriv ett annat program som vi kan använda för att hämta ett lösenord och visa det. Användaren ska mata in namnet på webbplatsen eller datorn som lösenordet gäller, och programmet ska skriva ut lösenordet.

Om filen inte går att öppna ska ett felmeddelande skrivas ut, och programmet ska avsluta. Om programmet inte hittar det sökta lösenordet ska det skriva ut **Finns inte**.