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 1 mars 2014

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 39. 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/2013-2014-p2/, senast lördag 22 mars 2014.
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) `0 + 1 + 2 - 3`

b) `0 + 1 - 2 / 3`

c) `0 + 1 - 2 + 3`

d) `0 * 1 * 2 + 3`

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 = 0; b = 1; c = 2; d = 3;
for (a = 0; a < 17000; ++a) {
    b = c;
}
a = 0;
while (a < d) {
    c = c + b;
    a = a + 1;
}
```

Uppgift 3 (2 p)

Anropet **rand()** till funktionen **rand** returnerar ett slumpmässigt heltal mellan **0** och **RAND_MAX**, inklusive gränserna. (**RAND_MAX** kan variera mellan olika system, och på min dator är det **2147483647**.) Ofta vill man ha ett tal mellan några andra gränser. Skriv funktionen **slumptal**, som tar argumentet **max**, och returnerar ett slumptal mellan **0** och **max**, inklusive gränserna. Till exempel ska anropet **slumptal(4)** returnera något av talen **0**, **1**, **2** och **3**. Funktionen **slumptal** ska anropa funktionen **rand**.

Uppgift 4 (3 p)

Skriv ett komplett C-program (med **#include** och allt) som läser in tre heltal. Därefter ska programmet slumpmässigt välja ett av dessa tre tal och skriva ut det.

Använd gärna funktionen **slumptal** från uppgiften ovan. Innan man kan använda den funktionen, bör slumptalsgeneratorn initieras, till exempel med anropet **srand(time(NULL));**

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.

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 5 (5 p)

Variablerna **i**, **j** och **n** är av typen **int**. Här är tio korta kodavsnitt med loopar. Hur många stjärnor skrivs ut i vart och ett av dessa kodavsnitt?

a)

```
for (i = 0; i < 5; ++i)
    printf("*");
```

b)

```
n = 5;
for (i = 0; i < n; i++)
    printf("*");
```

c)

```
n = 5;
for (i = 0; i < n - 1; ++i)
    printf("*");
```

d)

```
for (i = 0; i < 5; i++) {
    putchar('*');
}
```

e)

```
for (i = 0; i <= 3; ++i)
    printf("***");
```

f)

```
for (j = 0; j <= 5; j++)
    printf("*");
```

g)

```
for (i = 1; i <= 427; i++)
    printf("*");
```

h)

```
i = 0;
while (i < 6) {
    printf("*");
    i = i + 1;
}
```

i)

```
for (i = 0; i <= 3; ++i);
    printf("*");
```

j)

```
j = 0;
do
    printf("*");
while (j++ < 3);
```

Uppgift 6 (5 p)

Många amerikanska presidenter har namn av typen **Richard M. Nixon** och **George W. Bush**, dvs ett förnamn, en mellaninitial och ett efternamn. Här är en datatyp för att lagra sådana namn:

```
#define MAXNAMN 20

struct President {
    char fornamn[MAXNAMN + 1];
    char initial[1 + 1];
    char efternamn[MAXNAMN + 1];
};
```

a)

Skriv en funktion **namnbyte** som byter plats på förnamnet och efternamnet, till exempel så att **George W. Bush** får det nya namnet **Bush W. George**.

b)

Skriv en **main**-funktion som har en lokal variabel av typen **struct President**, som lägger in namnet **John F. Kennedy** i den, och sedan anropar funktionen **namnbyte** så att namnet i den lokala variabeln ändras till **Kennedy F. John**. Till slut ska enbart efternamnet (som alltså bör vara **John** om allt fungerat som det ska) skrivas ut.

Uppgift 7 (3 p)

Skriv ett program som definierar en array (även kallad vektor) som innehåller fem heltalselement. Programmet ska läsa in värden till alla element, och sedan skriva ut elementen i omvänd ordning.

Uppgift 8 (3 p)

När man lagrar reella tal i form av flyttal, antingen det är **float** eller **double**, går inte alla tal att lagra exakt. Därför kan det uppstå fel i beräkningarna, som ibland kan bli ganska stora. För att testa detta ska vi skriva ett program som adderar många flyttal:

Programmet ska läsa in ett tal (till exempel **2.1**), som vi kan kalla **x**, och ett antal gånger (till exempel **1000**), som vi kan kalla **n**, samt det förväntade resultatet om man adderar talet så många gånger (i det fallet **2100**). Därefter ska programmet räkna ut summan genom att addera talet **x n** gånger. (Man ska alltså inte bara multiplicera **x** och **n**.)

Till slut ska programmet skriva ut den beräknade summan, och skillnaden mellan den beräknade summan och det av användaren inmatade förväntade värdet.

Körexempel, med användarens inmatning understruken:

```
Talet x: 0.001
Antalet n: 1000000000
Förväntat resultat: 1000000
Beräknat resultat:: 1000000.018529
Skillnad: 0.018529
```

Uppgift 9 (5 p)

Skriv ett C-program (med **#include** och allt) som läser in rader, med ett ord på varje rad, ända tills användaren matar in något av orden **klar**, **sluta** och **avbryt**. Därefter ska programmet tala om dels hur många ord användaren skrev totalt (inklusive avslutningsordet), och dels antalet gånger användaren skrev något ord som börjar på bokstaven **A**. (Både **A** och **a** ska räknas.) Man kan förutsätta att användaren aldrig skriver några mellanslag, utan bara ett ord per rad.

I den här och alla andra uppgifter på tentan gäller: Normalt ska man aldrig använda funktionen **gets**, utan i stället till exempel **fgets**. Här kan du dock använda **gets**.

Uppgift 10 (5 p)

Textfilen **tal.txt** innehåller en stor mängd reella tal (till exempel **-12.34**), med ett tal per rad. Skriv ett C-program som läser den filen och talar om både hur många tal det finns totalt på filen, och hur många av talen som är större än medelvärdet. Om filen inte går att öppna, ska ett felmeddelande skrivas ut, och programmet ska avslutas.

Tips: Man kan behöva läsa filen två gånger: först en gång för att summera och räkna talen, så man kan beräkna medelvärdet, och sen en gång till för att räkna hur många av talen som är större än det medelvärdet.

Uppgift 11 (5 p)

Binärfilen **trianglar.bin** innehåller en stor mängd trianglar i form av den här posttypen:

```
struct Triangel {
    double sida1;
    double sida2;
    double sida3;
};
```

Skriv ett C-program som läser den filen och talar om hur många av trianglarna som är liksidiga, likbenta och med alla sidor olika. Man behöver inte kontrollera om trianglarna är omöjliga eller rätvinkliga, utan alla trianglar antas vara antingen liksidiga, likbenta eller med alla sidor olika. Om filen inte går att öppna, ska ett felmeddelande skrivas ut, och programmet ska avslutas.
