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

onsdag 4 juni 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 35. För godkänt betyg (3 respektive G) krävs 17 poäng.
Resultat och lösningar:	Meddelas via e-post eller på kursens hemsida, http://basen.oru.se/kurser/c/2013-2014-p2/, senast onsdag 25 juni 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. (Läraren är utomlands och går kanske inte att nå via telefon.)

- 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 + 1 * 1`

b) `1 + 1 / 1`

c) `(1 * 1) / 1`

d) `1 / (1 + 1)`

Uppgift 2 (2 p)

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

```
x = 1.1; y = 2.2; z = 3.3;
t = x + y;
while ( x < t) {
    x = x + 1;
}
if (x < t)
    z = z + 1;
else
    z = z - 1;
```

Uppgift 3 (2 p)

I en del länder mäter man längder i tum och fot. En tum är 2,54 centimeter.

Skriv ett komplett C-program (med **#include** och allt) som läser in en längd angiven i tum, och skriver ut den, men nu angiven i centimeter.

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: Man kan strunta i detaljer som bara behövs just när man utvecklar konsolprogram i Visual Studio, som konstiga teckenkoder för ÅÄÖ, och att fönstret med programkörningen försvinner när programmet avslutas.

Uppgift 4 (3 p)

Paracetamol är ett vanligt medel mot feber och smärta, som finns i många olika läkemedel, till exempel Alvedon och Panodil. I för stora mängder är paracetamol giftigt. Den högsta tillåtna dosen per dag beror på patientens kroppsvikt, och är (något förenklat) 60 mg (dvs 0,060 gram) paracetamol per kilo kroppsvikt. Det motsvarar 4,2 gram för en människa på 70 kg. En vanlig tablett innehåller 500 mg (dvs ett halvt gram) paracetamol.

Skriv ett komplett C-program (med **#include** och allt) som läser in kroppsvikten, och talar om hur många tabletter man får ta per dag.

Uppgift 5 (1 p)

Här är en funktion som är tänkt att sortera en array av heltal i stigande ordning, dvs med det minsta talet först.

```
/* Sorterar heltalsarrayen "a" med längden "length" i stigande ordning */
void sort_array(int a[], int length) {
    int sorted, in_rest, smallest_in_rest, temp;
    for (sorted = 0; sorted < length - 1; ++sorted) {
        smallest_in_rest = sorted;
        for (in_rest = sorted + 1; in_rest < length; ++in_rest) {
            if (a[in_rest] > a[smallest_in_rest]) {
                smallest_in_rest = in_rest;
            }
        }
        temp = a[sorted];
        a[sorted] = a[smallest_in_rest];
        a[smallest_in_rest] = temp;
    }
} /* sort_array */
```

Det finns ett fel i den. Vad är felet, och vad gör funktionen egentligen?

Uppgift 6 (3 p)

Skriv en **main**-funktion som först frågar efter ett antal tal, och sedan läser in så många heltal som angavs till en array. Därefter ska programmet sortera arrayen genom ett anrop till funktionen **sort_array** i uppgiften ovan. Till slut ska programmet skriva ut de sju minsta talen. Om antalet tal är mindre än sju, ska bara så många tal som finns skrivas ut.

> 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 7 (10 p)

Vi ska jobba med data om svenska tätorter. Här är en tabell:

Tätorter			
Kod	**Namn**	**Kommun**	**Folkmängd**
336	Stockholm	Stockholms kommun	1372565
6188	Örebro	Örebro kommun	107038
6076	Odensbacken	Örebro kommun	1374
6164	Åbytorp	Kumla kommun	755
....	....	....	....

Varje tätort har en unik kod i form av ett heltal, ett (inte nödvändigtvis unikt) namn, den ligger i en viss kommun, och den har en folkmängd. Det längsta namnet är 35 tecken långt

a) (2p)

Skapa datatypen **struct Tatort**, som ska användas för att lagra data om en tätort enligt ovan.

b) (1p)

Definiera en variabel som har datatypen **struct Tatort**, och initiera den med data om **Odensbacken** enligt exemplet ovan.

c) (2p)

Vi vill kunna visa data om tätorter på skärmen. Skriv därför funktionen **visa_tatort**, som tar en tätort (eller en pekare till en tätort, om du hellre vill det) som parameter, och skriver ut tätortens uppgifter, med lämpliga ledtexter, på standardutmatningen.

d) (3p)

Skriv funktionen **las_tatort**, som läser in data om en tätort. Funktionen ska skriva ut lämpliga ledtexter på standardutmatningen, läsa in data från standardinmatningen (som normalt är kopplad till tangentbordet), och på lämpligt sätt returnera den inlästa tätortens data till den anropande funktionen.

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**.

e) (2p)

Skriv en **main**-funktion som har två variabler av typen **struct Tatort**, som läser in data om två tätorter till dessa variabler med hjälp av funktionen **las_tatort**, och som slutligen använder **visa_tatort** för att skriva ut data om den av tätorterna som har det längsta namnet.

Uppgift 8 (3 p)

Vi vill skriva ut en "ruta" bestående av fem rader med tio stjärnor på varje rad. Här nedan är tre olika försök att göra detta.

Vad ger vart och ett av försöken för resultat?

a)

```
int rad = 0, kolumn = 0;
while (rad < 5) {
    while (kolumn < 10) {
        printf("*");
        ++kolumn;
    }
    printf("\n");
    ++rad;
}
```

b)

```
int rad, kolumn;
for (rad = 0; rad <= 5; ++rad) {
    for (kolumn = 0; kolumn <= 10; ++kolumn) {
        printf("*");
    }
    printf("\n");
}
```

c)

```
int rad, kolumn;
for (kolumn = 1; kolumn <= 5; ++kolumn) {
    for (rad = 1; rad <= 10; ++rad) {
        printf("*");
    }
    printf("\n");
}
```

Uppgift 9 (5 p)

Skriv ett C-program som läser heltal från standardinmatningen, ända tills användaren matar in talet noll, och sparar alla de jämna talen på en fil och alla de udda talen på en annan fil.

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

Uppgift 10 (5 p)

Skriv ett C-program som läser de två filerna med heltal från uppgiften ovan och skriver ut om det var flest jämna tal, flest udda tal, eller lika många.

Här behövs ingen felhantering.