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 28 februari 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.
Resultat och lösningar:	Meddelas via e-post senast lördag 21 mars 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) `4 - 3 + 2 - 1`

b) `4 - 2 * 3 + 1`

c) `4-2 * 3+1`

d) `4 * (3 - (2 - 1))`

Uppgift 2 (2 p)

Variabeln **a** är en array med tre heltal. Variablerna **b** och **c** är av typen **int**. Vilka värden har variablerna efter att följande kod har körts?

```
a[0] = 0;
a[1] = 0;
a[2] = 0;
b = a[1];
c = a[2] + b;
a[b] = 10;
for (b = 0; b < 3; ++b)
    a[b] = b * 2 + 1;
for (b = 0; b < 3; ++b)
    a[b] = b + 1;
for (b = 0; b < 70; ++b)
    c = c;
b = a[0] + a[1];
```

Uppgift 3 (4 p)

Regeringen har beslutat att förenkla reglerna för hur mycket skatt man ska betala. De nya reglerna är att inkomster under 100000 kronor per år är skattefria. För inkomster över 100000 kronor per år ska man betala hälften av den del som är större än 100000 kronor.

Exempel: En årsinkomst på 75000 kronor är skattefri. På en årsinkomst på 100002 kronor ska man betala 1 krona i skatt. På en årsinkomst på 200000 kronor ska man betala 50000 kronor i skatt.

Skriv ett komplett C-program (med **#include** och allt) som upprepat låter användaren mata in en årsinkomst, och skriver ut hur mycket skatten blir. Om inkomsten noll matas in, ska programmet omedelbart avslutas.

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, eller att fönstret med programkörningen försvinner när programmet avslutas.

Uppgift 4 (5 p)

Variablerna **i** och **j** 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 < 3; ++i) {
    printf("*");
}
```

b)

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

c)

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

d)

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

e)

```
i = 0;
while (i < 3) {
    printf("*");
    ++i;
}
```

f)

```
for (i = 0; i < 43*1000; i++)
    printf("*");
```

g)

```
i = 7;
do {
    i = 0;
    printf("*");
} while (i > 0);
```

h)

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

i)

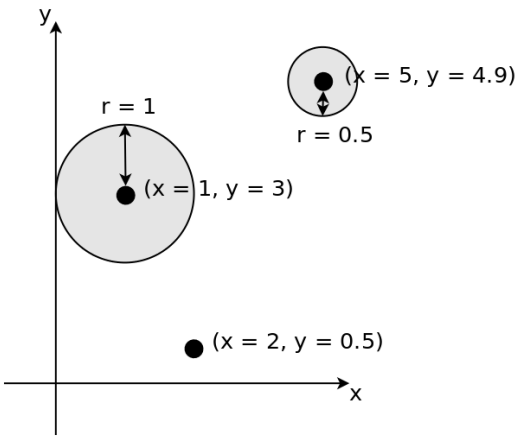
```
for (j = 0; j < 3; ++j) { ++i; printf("*"); }
```

j)

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

Uppgift 5 (10 p)

Här är ett koordinatsystem med några punkter och ett par cirklar.



a) Skapa posttypen **struct Cirkel**, som beskriver en cirkel. Vi behöver lagra x- och y-koordinaterna för mittpunkten, och radien.

b) Definiera en variabel av typen **struct Cirkel** och initiera den med data om den stora cirkeln i figuren i uppgiften ovan.

c) Skriv funktionen **area**. Funktionen ska ta en cirkelpost (eller pekare till den) som argument och returnera dess area. Arean A av en cirkel ges av denna formel. π är ungefär 3.14159265358979323846.

$$A = \pi \cdot r^2$$

d) Skriv funktionen **centrumavstand** som tar två cirkelposter (eller pekare till dem) som argument, och returnerar avståndet mellan cirklarnas mittpunkter. Avståndet d mellan de två punkterna (x_1, y_1) och (x_2, y_2) kan beräknas med Pythagoras sats:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

d) Skriv funktionen **avstand** som tar två cirkelposter (eller pekare till dem) som argument, och returnerar avståndet mellan cirklarnas ytterkanter. Gör det genom att anropa funktionen **centrumavstand**, och sedan subtrahera de två cirklarnas radier.

e) Vi vill provköra de tre funktionerna från deluppgifterna ovan. Skriv därför en **main**-funktion, som anropar dem med lämpliga argument, och skriver ut resultaten.

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 funktionen **visa_array** som tar en array med heltal som argument, och som skriver ut talen i arrayen på skärmen med 10 tal per rad. Om det behövs fler argument till funktionen än bara heltalsarrayen, så ha med dem också.

b) Skriv en **main**-funktion som först låter användaren mata in 47 heltal till en array, och därefter använder funktionen **visa_array** för att skriva ut dem.

Uppgift 7 (5 p)

Skriv ett program som upprepat läser in kommandon från användaren. Beroende på vilket kommando som användaren gav, ska programmet göra olika saker:

- Om kommandot är **avsluta** eller **quit** ska programmet avslutas.
- Om kommandot är **stjärnor** ska programmet fråga efter antalet stjärnor (ett heltal) och sedan skriva ut så många stjärnor som talet anger.
- Om kommandot är **plus** ska programmet fråga efter två flyttal, och sedan skriva ut deras summa.
- Om kommandot är något annat ska ett felmeddelande skrivas ut.

Uppgift 8 (8 p)

a) Skriv ett program som slumpar fram data om ett slumpmässigt antal cirklar med slumpmässiga (men rimliga) data, och sparar dessa data på en fil.

Man kan använda anropet **rand()** för att få ett stort, slumpmässigt heltal. Innan man använder den funktionen bör slumptalsgeneratorn initieras, till exempel med anropet **srand(time(NULL));**

b) Skriv ett annat program som läser filen ovan, och talar om vilken cirkel som har störst area, och hur stor den arean är.

För både a- och b-uppgiften gäller att om filen inte går att öppna ska ett felmeddelande skrivas ut, och programmet ska avslutas.
