Örebro universitet
Institutionen för naturvetenskap och teknik
Thomas Padron-McCarthy (thomas.padron-mccarthy@oru.se)

Tentamen i distanskursen

Programmering C

torsdag 16 mars 2017

Gäller som tentamen för:
DT104G Programmering C, provkod 0100

(Campuskursen "Programmering grundkurs" har en egen tenta.)

Hjälpmedel:	Ordbok för översättning.
Poängkrav:	Maximal poäng är 40 . För godkänt betyg (3 respektive G) krävs 20 poäng.
Resultat och lösningar:	Meddelas via e-post senast torsdag 6 april 2017.
Återlämning av tentor:	Efter att resultatet meddelats kan tentorna hämtas elektroniskt via Studentforum.
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 (3 p)

Vad skrivs ut när vi kör följande C-program?

```
#include <stdio.h>

int main(void) {
    int a = 0, b = 1, c, d;

    while (a < b) {
        if (a < 2)
            a = a + 2;
        else
            a = a + 3;
        for (c = 0; c < 2; ++c)
            b = b + 1;
            printf("b = %d\n", b);
    }
    printf("a = %d\n", a);
    printf("d = %d\n", d);

    return 0;
}
```

En del variabler kan ha ett odefinierat värde ("skräp") när de ska skrivas ut. Ange då detta!

Uppgift 3 (2 p)

Skriv en C-funktion som heter **value_is_ok**, som tar ett mätvärde som parameter, och som returnerar ett sant värde om mätvärdet ligger mellan 3.5 och 4.5. Annars ska ett falskt värde returneras.

Uppgift 4 (4 p)

a) (3p) Vi vill generalisera funktionen i uppgiften ovan, så nu ska vi skriva en funktion som heter **is_within_bounds** och tar tre parametrar: ett mätvärde, ett nominellt värde, och en felgräns. Alla är flyttal. Funktionen ska returnera ett sant värde om mätvärdet avviker från det nominella värdet med högst felgränsen. Annars ska ett falskt värde returneras.

b) (1p) Gör en ny implementation av funktionen **value_is_ok** som anropar funktionen **is_within_bounds**. (Det nominella värdet är 4, och felgränsen 0.5.)

I den här och alla andra uppgifter på tentan gäller: Om du ska anropa en funktion från en tidigare uppgift, behöver du inte skriva koden för den funktionen på nytt. Du får också anropa funktionen även om du inte gjort uppgiften där man skulle skriva den.

Uppgift 5 (4 p)

Skriv en **main**-funktion som upprepat läser in ett mätvärde, ett nominellt värde, och en felgräns, anropar funktionen **is_within_bounds** med de inmatade värdena, och skriver ut om mätvärdet låg mellan felgränserna eller inte.

Programmet ska avslutas direkt när man matar in noll som alla tre talen.

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 6 (6 p)

a) (3p) Skriv funktionen **antal_under_medel**, som tar två parametrar: en array med flyttal, och ett heltal som anger antalet tal i arrayen. Funktionen ska returnera antalet tal i arrayen som är mindre än medelvärdet av talen. (Ett tips är att man kan behöva gå igenom arrayen två gånger: en gång för att beräkna medelvärdet, och en gång för att räkna antalet tal som är mindre än medelvärdet.)

b) (3p) Skriv en **main**-funktion som först frågar efter antalet tal i arrayen, sen läser in så många tal till en array, och därefter skriver ut antalet tal som var mindre än medelvärdet av talen. **main**-funktionen ska anropa **antal_under_medel**, och arrayen ska vara en lokal variabel i **main** med plats för 100 tal. Om användaren matar in ett orimligt antal, dvs mindre än ett eller större än 100, ska ett felmeddelande skrivas ut, och programmet ska avslutas.

Uppgift 7 (3 p)

Vi vill kunna beräkna följande uttryck:

$$\frac{x^2}{1 - z\sqrt{x^4 - y^5}}$$

Skriv därför ett komplett C-program (med **#include** och allt) som läser in flyttalsvärden på **x**, **y** och **z**, och skriver ut värdet av uttrycket. Om deluttrycket i roten är mindre än noll, eller om nämnaren i divisionen är lika med noll, går uttrycket inte att beräkna. I så fall ska programmet inte försöka beräkna uttrycket, utan det ska skriva ut att uttrycket inte går att beräkna.

Uppgift 8 (11 p)

Här är en datatyp för att lagra geometriska figurer:

```
#define MAX_ANTAL_SIDOR 20

struct President {
    int antal_sidor;
    double sidlangder[MAX_ANTAL_SIDOR];
};
```

a) (4p) Vi kommer ihåg från inlämningsuppgift ett i kursen att en triangel är omöjlig om en av sidorna är längre än summan av de två andra. Det gäller alla figurer i planet, även med fler än tre sidor: Om en av sidorna är längre än summan av alla de andra sidorna, är figuren omöjlig. Skriv funktionen **mojlig_figur**, som tar en figurpost (eller, om du vill, en pekare till den) som parameter, och returnerar ett sant värde om figuren är möjlig att rita. Om den är omöjlig, ska ett falskt värde returneras.

b) (2p) Egentligen vill vi förstås rita upp figurerna på skärmen, men tills vidare nöjer vi oss med att skriva ut figurposternas innehåll. Skriv därför funktionen **visa_figur**, som skriver ut alla sidlängderna för en figur. Funktionen ska ta en figurpost (eller, om du vill, en pekare till den) som parameter. Om figuren är omöjlig, vilket avgörs med ett anrop till funktionen ovan, ska en varning skrivas ut om att figuren är omöjlig.

c) (3p) Skriv en funktion som heter **las_figur**, och som läser in data om en figur. Det ska gå att mata in figurer med tre till MAX_ANTAL_SIDOR stycken sidor. Funktionen ska skriva ut lämpliga ledtexter på standardutmatningen, och läsa in data från standardinmatningen (som normalt är kopplad till tangentbordet). Den inlästa posten ska returneras som returvärde från funktionen, eller via en pekarparameter.

d) (2p) Skriv en **main**-funktion som har två lokala variabler av typen **struct Figur**, och som läser in data om två figurer till de variablerna med hjälp av funktionen **las_figur**. Därefter ska programmet visa båda figurerna med hjälp av funktionen **visa_figur**.

Uppgift 9 (6 p)

En textfil med namnet **figurer.txt** innehåller data om figurer, med en figur per rad:

```
3 2.0 9.1 9.1
5 1.0 2.0 1.0 2.0 34.5
4 0.4 1.62 10.3 9.2
```

Exemplet innehåller en triangel, en omöjlig femhörning och en fyrhörning.

Skriv ett C-program som läser figurerna från filen och skriver de figurer som är möjliga på en fil som heter **möjliga.txt**, med samma format som indatafilen. Funktionen **mojlig_figur** ska användas för att avgöra om figurerna är möjliga.

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