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

Tentamen i distanskursen

Programmering C

onsdag 1 juni 2016

Gäller som tentamen för:
DT104G Programmering C, provkod 0100
DT1006 Datateknik A, Programmering C, distans, 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 onsdag 22 juni 2016.
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) `2 - 3 - 4 + 5`

b) `2 * 3 / 4 + 5`

c) `2 % 3 % 4 + 5`

d) `2+3 / 4+5`

Uppgift 2 (3 p)

Här är ett C-program:

```
#include <stdio.h>

int main(void) {
    int a, b, c, i;

    printf("Ange ett heltal: ");
    scanf("%d", &a);

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

    return 0;
}
```

Vad skrivs ut av programmet om användaren matar in

a) 0

b) 5

c) 1000

Uppgift 3 (2 p)

Skriv ett komplett C-program (med **#include** och allt) som skriver ut alla heltal mellan 1 och 100, och respektive tals kvadrat och kvadratrot. De första raderna i utskriften kan se ut så här:

```
1 1 1.000000
2 4 1.414214
3 9 1.732051
4 16 2.000000
5 25 2.236068
```

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 (3 p)

Vi vill kunna beräkna följande uttryck:

$$\frac{1}{1-\pi x}\left(\sqrt{\pi x^2}-\sqrt{\pi y^2}-1\right)$$

π är ungefär lika med 3.14159265358979323846. Vi kan anta att include-filen **pi.h** innehåller makrot **PI**, som är definierat till det värdet.

Skriv därför ett komplett C-program (med **#include** och allt) som låter användaren mata in **x** och **y** (två flyttal), och sedan skriver ut värdet av uttrycket. 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 i stället skriva ut ett felmeddelande.

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 (7 p)

a)

Skriv C-funktionen **maxabs** som tar två flyttal som argument och returnerar det som har det största absolutbeloppet, dvs är "längst bort från noll". Exempelvis har 4.3 större absolutbelopp än -2.7, och -2.7 har större absolutbelopp än 2.5.

b)

Skriv funktionen **maxabs_array**, som har det här funktionshuvudet:

```
void maxabs_array(float a[], float b[], float c[], int n);
```

De tre arrayerna **a**, **b** och **c** är alla lika stora, med en storlek som anges av **n**. **maxabs_array** ska gå igenom arrayerna **a** och **b**, anropa **maxabs** med varje par av tal som finns på samma position i **a** och **b**, och placera resultatet på motsvarande position i **c**.

c)

Skriv en **main**-funktion för att provköra **maxabs_array**. Användaren ska mata in en arraystorlek (högst 100), sedan flyttal till två arrayer, och till sist ska **main**-funktionen anropa **maxabs_array** och skriva ut resultatet.

Om man matar in en otillåten arraystorlek (mindre än noll eller större än 100) ska ett felmeddelande skrivas ut, och programmet ska avslutas.

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 6 (11 p)

Nu ska vi redigera film. Man kan "klippa ihop" en film av flera kortare filmklipp, och vi behöver en posttyp (med C-terminologi en **struct**) som kan användas för att representera ett filmklipp. Det som ska finnas med i posten är:

- Filmklippets längd i hela minuter (ett heltal) och sekunder (ett flyttal).
- En kort text (högst 30 tecken) som beskriver filmklippets innehåll.

a)

Definiera posttypen. Den ska heta **struct Filmklipp**.

b)

Definiera en variabel av typen **struct Filmklipp** och initiera den med data om ett filmklipp med längden **2** minuter och **16.4** sekunder, och innehållet **Thomas pratar**.

c)

Skriv en funktion som heter **visa_filmklipp**, som skriver ut data om ett filmklipp på skärmen. Funktionen tar filmklipposten som parameter, och skriver ut den. Ett förslag på hur utskriften skulle kunna se ut:

```
Innehåll: Thomas pratar
Tid: 2 min 16.4 s
```

d)

Skriv en funktion som heter **las_filmklipp**, och som läser in data om ett filmklipp. Funktionen ska skriva ut lämpliga ledtexter på standardutmatningen, och läsa in data från standardinmatningen (som normalt är kopplad till tangentbordet).

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**. (Därmed inte sagt att någon av dessa funktioner nödvändigtvis är det bästa valet i den här uppgiften.)

e)

Skriv en funktion som heter **klipp_ihop**, som tar två filmklipp som argument, och som returnerar ett filmklipp där de båda är hopklippta. Beskrivningen av innehållet ska vara **Hopklippt film**. Tiden blir summan av de två filmklippens tider, men tänk på att justera sekunderna ifall de blir 60 eller mer. Exempelvis ska ett filmklipp på 1 min 40 s och ett annat på 1 min 30 s inte resultera i ett filmklipp på 2 min 70 s, utan ett på 3 min 10 s.

f)

Skriv en **main**-funktion som läser in data om två filmklipp till två lokala variabler med **las_filmklipp**, slår ihop dem med **klipp_ihop**, och skriver ut resultatet med **visa_filmklipp**.

Uppgift 7 (8 p)

a)

Skriv ett C-program som läser in data om filmklipp, (som i uppgiften ovan) med **las_filmklipp** och sparar dem på en fil. Välj själv formatet på filen, och hur inmatningen ska avslutas.

b)

Skriv ett C-program som läser filen med filmklipp, klipper ihop alla filmklippen till en komplett film, och skriver ut längden på den hopklippta filmen. Man kan använda **klipp_ihop** och **visa_filmklipp**, men kanske inte **las_filmklipp** (eftersom den läser från standardinmatningen och inte från en fil).

Uppgift 8 (5 p)

Skriv ett program som läser in personnummer på formen **ÅÅMMDD-SSSS**, med ett personnummer per rad, upprepat tills användaren ger en tom rad.

Därefter ska programmet ange hur många personer som är män och hur många som är kvinnor. Om den näst sista siffran i personnumret är jämn, är personen en kvinna, och om udda, man.

Exempel på hur en programkörning skulle kunna se ut (med användarens inmatning understruken):

```
Ange personnummer (ett per rad, avsluta med tom rad):
631211-1658
451019-2270
811214-3360
821001-6720
851001-2273

Antal kvinnor: 2
Antal män: 3
```