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

torsdag 4 juni 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 senast torsdag 25 juni 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) `1 + 2 * 3 - 4`

b) `1+2 * 3-4`

c) `1 + 2 * 3 / 4`

d) `1 + 2.0 * 3 / 4`

Uppgift 2 (2 p)

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

```
a = 1;
b = 2;
x = 1 + 2;
z = a;

while (x < 5) {
    a = a + b;
    if (a == 3)
        y = x + 0.5;
    else
        x = x + 3;
}
++b;
```

Uppgift 3 (4 p)

Regeringen har ändrat reglerna för hur mycket skatt man ska betala. De nya reglerna är att den skatt man ska betala anges av följande formel, där variabeln **x** anger årsinkomsten i kronor, och **π** är ungefär 3.14159265358979323846:

$$\frac{x}{2}+\frac{1}{2}\cdot\sin\left(\frac{x}{\pi}\right)\cdot\sqrt{x+1000}$$

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)

Skriv ett program som läser in två positiva hela tal och som skriver ut alla tal från och med det minsta inlästa talet till och med det största inlästa talet, och som därefter skriver ut summan av alla utskrivna tal. Två körexempel, där användarens inmatning är understruken:

```
Ge första talet: 21
Ge sista talet: 25
21 + 22 + 23 + 24 + 25 = 115

Ge första talet: 11
Ge sista talet: 5
5 + 6 + 7 + 8 + 9 + 10 + 11 = 56
```

Uppgift 5 (5 p)

Skriv ett program som börjar med att läsa in data till en array av flyttal med 5 element. När hela arrayen lästs in ska alla elementen i arrayen ökas med ett inläst procentvärde, varefter de nya arrayelementen skrivs ut.

Körexempel, där användarens inmatning är understruken:

```
Ge fem reella tal: 2.3 4.5 3.2 1.2 5.6
Ge ökning i procent: 10.0
Ny array: 2.53 4.95 3.52 1.32 6.16
```

Uppgift 6 (4 p)

a) Skriv funktionen **rita_fyrkant**, som tar två heltalsargument **r** och **k**, och som skriver ut en fyrkant på skärmen bestående av **r** stycken rader med vardera **k** stycken stjärnor.

b) Skriv en **main**-funktion som frågar efter och läser in ett antal rader och ett antal kolumner, och sedan ritar ut en sådan fyrkant genom att anropa funktionen **rita_fyrkant**.

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.

Scenario till de följande uppgifterna

Rymdpolisen har till uppgift att patrullera rymden. Då och då får de ett larm från någon planet i galaxen, och rycker ut med sina rymdskepp.

Rymdpolisen skriver upp sina utryckningar på en fil, **utryckningar.txt**. Varje rad i filen beskriver en utryckning, och består av ett datum, ett klockslag, namnet på den planet som utryckningen gällde, samt vad det handlade om. Exempel:

```
2210-08-12 18:04 Saturnus Fortkörning
2210-08-12 18:07 Glorbatron Inbrott
2210-08-13 01:33 Saturnus Bråkiga rymdmonster
2210-08-13 02:19 Zoflomork Fortkörning
```

I exemplet ovan har vi haft två utryckningar till planeten Saturnus, och en var till planeterna Glorbatron och Zoflomork.

Planetnamn består av ett enda ord med högst 20 tecken. Planeternas namn är unika. Beskrivningen av vad utryckningen handlade om kan bestå av flera ord, men högst 30 tecken. Det kan finnas upp till tusen planeter. Filen kan innehålla flera miljoner utryckningar.

Uppgift 7 (10 p)

a) Skapa posttypen **struct Utryckning**, som beskriver en utryckning. Vi behöver lagra datumet, klockslaget, namnet på planeten, och vad utryckningen handlade om.

b) Definiera en variabel av typen **struct Utryckning** och initiera den med data om den första utryckningen i exempelfilen ovan, den om fortkörning vid Saturnus.

c) Vi vill kunna visa utryckning-posternas innehåll på skärmen. Skriv en funktion som heter **visa_utryckning**, och som skriver ut data om en utryckning. Funktionen ska ta en utryckningspost (eller, om du vill, en pekare till den) som parameter. Här är ett exempel på hur utskriften kan se ut:

```
Datum: 2210-08-12
Tid: 18:04
Plats: Saturnus
Angående: Fortkörning
```

d) Skriv en funktion som heter **las_utryckning**, och som läser in data om en utryckning. 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 pekare.

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) Skriv en **main**-funktion som har två lokala variabler av typen **struct Utryckning**, och som läser in data om två utryckningar till de variablerna med hjälp av funktionen **las_utryckning**. Därefter ska programmet visa båda utryckningarna med hjälp av funktionen **visa_utryckning**.

Uppgift 8 (3 p)

Som nämndes i scenariot skriver rymdpolisen upp sina utryckningar i filen **utryckningar.txt**. Vi vill kunna lägga till en ny utryckning sist i filen. Skriv ett C-program som läser in data om en utryckning med hjälp av funktionen **las_utryckning**, och lägger till den sist i filen.

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

Ett tips: Om man anropar **fopen** med **mode**-strängen **"a"**, öppnas filen för skrivning sist i filen.

Uppgift 9 (6 p)

Rymdpolisen vill sammanställa statistik över brottsligheten i rymden. Skriv därför ett program som läser in namnet på en planet, läser filen **utryckningar.txt**, och talar om hur många utryckningar som skett till den angivna planeten.

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