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 21 augusti 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 40. För godkänt betyg (3 respektive G) krävs 20 poäng.
Resultat och lösningar:	Meddelas via e-post eller på kursens hemsida, http://basen.oru.se/kurser/c/2013-2014-p2/, senast torsdag 11 september 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 + 2 * 3`

b) `1 - 2 - 3`

c) `1 + 2 / 3`

d) `1 + (2 / 3)`

Uppgift 2 (2 p)

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

```
x = 1; y = 2; z = 3;
for (x = 0; x < 1000; ++x)
    y = z;
x = 1;
while (x < 3) {
    x = x + 1;
    z = x;
}
```

Uppgift 3 (3 p)

Skriv ett komplett C-program (med **#include** och allt) som först läser in ett **gräns** i form av ett heltal, och sen läser in heltal ända tills man matar in ett tal som är lika med gränsen. Därefter ska programmet skriva ut hur många av de inmatade talen som var mindre än gränsen och hur många som var större.

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 (3 p)

Skriv funktionen **replace_char**, som byter ut alla förekomster av ett visst tecken i en sträng mot ett annat tecken. Exempelvis ska anropet

```
replace_char("abraham", 'a', 'x')
```

ge "xbrxhxm" som resultat.

```
replace_char("ABCabcdd", 'a', '.')
```

ska ge "ABC.bcdd" som resultat. 'A' och 'a' räknas alltså som olika tecken.

Uppgift 5 (3 p)

Skriv en **main**-funktion som först läser in en sträng, därefter två tecken, sedan anropar **replace_char** för att byta ut det först angivna tecknet mot det andra i den inmatade strängen, och sedan skriver ut resultatet.

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.

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**.

Uppgift 6 (6 p)

Du har just köpt en C-kompilator från Lågprisboden. Tyvärr saknas några viktiga funktioner i standardbiblioteket, som du nu måste skriva själv. Skriv alltså följande funktioner. De ska fungera likadant som standardfunktionerna. Du får gärna göra anrop till andra standardfunktioner.

a) (2p) **strlen**

b) (3p) **gets**

c) (1p) **getchar**

Uppgift 7 (7 p)

Spelet som heter "tic-tac-toe" på engelska brukar översättas med "luffarschack" på svenska, men luffarschack spelas egentligen på en större spelplan än de 9 rutor som tic-tac-toe använder, och man ska ha fem i rad, inte tre som i tic-tac-toe.

I ett program som ska kunna spela luffarschack behöver man hålla reda på spelplanen, och kunna skriva ut den. Ett sätt att lagra spelplanen är som en array av tecken, där tecknen 'x' och 'o' betyder kryss och ringar, och tecknet '.' betyder tom ruta.

a) (2p) Gör de definitioner som behövs för att lagra spelplanen. Gör planen ordentligt stor, till exempel 50x50 rutor.

b) (5p) Skriv en C-funktion som skriver ut spelplanen! Bara den del av planen som används ska skrivas ut.

Exempel: Om hela planen är 20x50 rutor, och bara innehåller fyra markeringar i mitten, vill man inte att den ska skrivas ut så här:

```
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
......................xx..........................
.......................o..........................
........................o.........................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
..................................................
```

Skriv i stället bara ut den intressanta biten i mitten:

```
xx.
.o.
..o
```

Uppgift 8 (5 p)

Binärfilen **datafil.bin** innehåller en massa heltal, och nu vill man veta hur många av dessa heltal som ligger i ett visst intervall.

Skriv ett fullständigt C-program som låter användaren mata in två heltal, och som sen öppnar och läser binärfilen, och slutligen skriver ut hur många tal som låg i intervallet mellan (och inklusive) de två inmatade talen.

Antag att filen "datafil.bin" innehåller följande heltal:

```
3  5  20  12  24  -7  12  8  0  34  -14
```

Då kan programkörningen till exempel se ut så här, med användarens inmatning understruken:

```
Skriv nedre gränsen (ett heltal): 11
Skriv övre gränsen (ett heltal): 20
Filen innehöll 3 tal mellan 11 och 20.
```

Den enda felkontroll du behöver göra är att kontrollera att filen gick att öppna, och annars skriva ut ett felmeddelande och avsluta programmet.

Uppgift 9 (5 p)

En textfil som heter **ord.txt** innehåller en massa ord.

Skriv ett fullständigt C-program som läser textfilen, och som sen skriver ut orden i omvänd ordning. Med ett "ord" menar vi en följd av icke-blanka tecken, som omges av blanka tecken, eller av filens början eller slut. (Det betyder att du kan använda fscanf med argumentet "%s" för att läsa orden!)

Inga ord är längre än 15 bokstäver. Det finns högst 100 ord i filen. Programmet behöver inte bry sig om att dela upp utskriften i rader.

Den enda felkontroll du behöver göra är att kontrollera att filen gick att öppna, och annars skriva ut ett felmeddelande och avsluta programmet.

Exempel: Om filen innehåller texten:

```
Visst gör det ont när kroppar brister,
men inte stoppar det Arnold inte!
```

kan det ge utskriften:

```
inte! Arnold det stoppar inte men brister, kroppar när ont det gör Visst
```

Uppgift 10 (5 p)

Läraren är lat och vill inte rätta tentor. Skriv därför ett program som slumpar fram tentaresultat, så läraren slipper rätta uppgifterna.

Man ska först mata in maxpoängen på de olika frågorna, och avsluta med noll. Alla maxpoängen är heltal. Därefter ska programmet slumpa fram poängen på varje enskild uppgift. Poängen på en uppgift kan variera från noll till maxpoängen på uppgiften. Det är bara hela poäng, inga decimaler. Dessutom ska programmet skriva ut summan av de framslumpade poängen. Här är ett exempel på hur en programkörning skulle kunna se ut, med användarens inmatning understruken:

```
Ange poängen på uppgifterna. Avsluta med talet noll.
1
2
3
3
3
6
7
5
5
5
0
0 0 2 3 0 5 1 4 3 5
Summa: 23
```

Tentan kan inte ha mer än tjugo uppgifter.

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