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

Tentamen i

Programmering i språket C

för D1 m fl, även distanskursen

torsdag 23 augusti 2012

Gäller som tentamen för:
DT1016 Datateknik A, Programmering grundkurs, provkod 0100
DT1007 Datateknik A, Tillämpad datavetenskap, provkod 0410
DT1006 Datateknik A, Programmering C, distans, provkod 0100

Hjälpmedel:	Inga hjälpmedel.
Poängkrav:	Maximal poäng är 36. För godkänt betyg (3 respektive G) krävs 18 poäng.
Resultat och lösningar:	Meddelas via e-post eller på kursens hemsida, http://basen.oru.se/kurser/c/2011-2012-p2/, senast torsdag 13 september 2012.
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. (Gäller inte om du skriver tentan på annan ort.)
- 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);
```

Uppgift 1 (1 p)

Vilka värden har följande C-uttryck? (Det kan hända att värdet inte går att avgöra från den information som givits. Ange i så fall detta.)

a) 1 + 2 - 1 + 2

b) 1 * 2 - 1 * 2

c) 1 < 2 || x < y

d) 1 < 2 && x < y

Uppgift 2 (1 p)

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

```
x = 1;
y = 2;
z = 3;
z = x + y;
if (x > y) {
    z = x + y;
}
else if (y < z) {
    printf("x = 17");
    if (z < 4)
        x = x + y;
}
else if (x > y) {
    z = y;
}
else {
    x = x - y;
}
```

Uppgift 3 (4 p)

Skriv ett komplett C-program (med **#include** och allt) som först låter användaren skriva in tre heltal: ett antal, en nedre gräns, och en övre gräns. Därefter ska programmet läsa in ytterligare så många heltal från användaren som angavs av antalet. När alla de heltalen är inmatade, ska programmet skriva ut **Ok** om alla heltalen låg inom det angivna intervallet. Annars ska programmet skriva ut **Inte Ok**. Ett tal räknas som inom intervallet om det är större än eller lika med den nedre gränsen, och mindre än eller lika med den övre gränsen.

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, och att fönstret med programkörningen försvinner när programmet avslutas.

Uppgift 4 (4 p)

a)

Skriv en funktion, **tre_lika**, som tar tre flyttal som argument, och som returnerar ett sant värde om alla tre talen är lika. Annars ska funktionen returnera ett falskt värde.

b)

Skriv en annan funktion, **tre_olika**, som tar tre flyttal som argument, och som returnerar ett sant värde om alla tre talen är olika. Om två eller fler av talen är lika, ska funktionen returnera ett falskt värde.

c)

Vi vill provköra de två funktionerna **tre_lika** och **tre_olika**. Skriv därför en **main**-funktion som läser in tre tal, anropar **tre_lika** och **tre_olika** med de talen som argument, och 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 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 5 (1 p)

Vi ska skriva ett program som hanterar termosar. Skapa därför posttypen **Termos**. En termos har ett typnamn, som är en textsträng, till exempel **Sahara MWS-A500**. Den har också en **volym**, som är ett flyttal, till exempel **0.5**. Dessutom har den ett innehåll, som också anges med en textsträng, till exempel **kaffe**, och en innehållsvolym, som anger hur mycket som faktiskt finns i termosen. Innehållsvolymen är ett flyttal, till exempel **0.3**.

Uppgift 6 (1 p)

Definiera en variabel av typen **struct Termos** och initiera den med data om en termos av typen **Sahara MWS-A500**, med en volym på **0.5** liter, som innehåller **0.3** liter **kaffe**.

Uppgift 7 (2 p)

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

```
Typ: Sahara MWS-A500
Volym: 0.50
Innehåll: kaffe
Innehållets volym: 0.30
```

Uppgift 8 (3 p)

Skriv en funktion som heter **las_termos**, och som läser in data om en termos. 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**.

Uppgift 9 (1 p)

Varför ska man aldrig använda funktionen **gets**? Förklara!

Uppgift 10 (1 p)

Skriv funktionen **rimlig_termos**, som gör en rimlighetskontroll av data om en termos. Den ska bara kontrollera att innehållets volym inte är större än termosens volym. Om termosens data är rimliga, ska funktionen returnera ett sant värde. Annars ska den returnera ett falskt värde.

Uppgift 11 (2 p)

Skriv en **main**-funktion som har en lokal variabel av typen **struct Termos**, och som läser in data om en termos till den variabeln med hjälp av funktionen **las_termos**. Därefter ska programmet visa dessa data med hjälp av funktionen **visa_termos**. Om termosens data är orimliga (använd funktionen **rimlig_termos**) ska det också skriva ut en varning om det.

Uppgift 12 (2 p)

Skriv funktionen **volymsumma**, som returnerar summan av volymen av alla termosarna i en array med termosar. Förutom själva arrayen ska funktionen ta ett heltal som argument, som anger hur många termosar som finns i arrayen.

Uppgift 13 (3 p)

Vi vill provköra funktionen **volymsumma**. Skriv därför en **main**-funktion som anropar funktionen med lämpliga data, och kontrollerar att den ger det förväntade svaret. (För en bra testning på riktigt skulle man behöva fler anrop, men här nöjer vi oss med ett enda.) Om funktionen ger fel svar, ska ett tydligt felmeddelande skrivas ut. Annars behöver inget skrivas ut.

Uppgift 14 (5 p)

Skriv ett C-program som läser in data om termosar med hjälp av funktionen **las_termos**, och sparar dem på en fil. Inmatningen ska avslutas när man matat in en orimlig termos. (Den orimliga termosen ska inte sparas på filen.) Använd funktionen **rimlig_termos**. Om filen inte går att öppna för skrivning, ska ett felmeddelande skrivas ut, och programmet ska avslutas.

Uppgift 15 (5 p)

Skriv ett C-program som först frågar efter en volym, och sedan skriver ut alla termosar på filen från uppgiften ovan som är minst så stora. Använd funktionen **visa_termos**. Om filen inte går att öppna för läsning, ska ett felmeddelande skrivas ut, och programmet ska avslutas.
