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 22 augusti 2013

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
DT1016 Datateknik A, Programmering grundkurs, provkod 0100
DT1007 Datateknik A, Tillämpad datavetenskap, provkod 0410

(There is also an English version of this exam.)

---

| | |
|---|---|
| **Hjälpmedel:** | Inga hjälpmedel. |
| **Poängkrav:** | Maximal poäng är 40.<br>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/2012-2013-p2/, senast torsdag 12 september 2013. |
| **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);
```

# Uppgift 1 (1 p)

Vilka värden har följande uttryck i C?

a) 4 * 3 − 2 + 1

b) 4 − 3*2 + 1

c) 4 / 3 − 4 / 2

# Uppgift 2 (3 p)

Vad skrivs ut när följande C-program körs?

```
#include <stdio.h>

int g(int x, int y) {
    int i, a, b, c;
    a = 17;
    b = 2;
    c = 38;
    for (i = x; i < y; i++)
        printf("*");
    return x + y;
}

int main(void) {
    int a;
    int b, c;
    for (a = 1; a < 3; ++a) {
        b = a + 1;
        c = g(a, b);
        printf("a = %d, b = %d, c = %d\n", a, b, c);
    }
    return 0;
}
```

# Uppgift 3 (3 p)

Skriv ett komplett C-program (med **#include** och allt) som upprepat läser in tre heltal och skriver ut deras medelvärde (med decimaler). Programmet ska läsa in tre heltal ända tills användaren matar in tre tal som alla är lika med noll.

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.

# Scenario

Här är en posttyp, **struct Triangel**, som vi ska använda för att lagra data om trianglar:

```
struct Triangel {
    double a, b, c;
};
```

**a**, **b** och **c** är längderna på triangelns sidor. Sidorna kan vara lagrade i vilken ordning som helst, så till exempel finns det ingen garanti att triangelns längsta sida är någon specifik av **a**, **b** och **c**. Alla längder ska vara större än noll.

# Uppgift 4 (1 p)

Definiera en variabel av typen **struct Triangel** och initiera den med data om triangeln med sidorna **3**, **4** och **3.5**.

# Uppgift 5 (2 p)

Skriv en funktion som heter **visa_triangel**, och som visar en triangels data på skärmen. Den ska alltså skriva ut triangelns uppgifter, med lämpliga ledtexter, på standardutmatningen. Du får själv välja om du vill att funktionshuvudet ska se ut så här:

```
void visa_triangel(struct Triangel t)
```

eller så här:

```
void visa_triangel(struct Triangel *tp)
```

# Uppgift 6 (3 p)

Skriv en funktion som heter **las_triangel**, och som läser in en triangels data. Funktionen ska skriva ut lämpliga ledtexter på standardutmatningen, och läsa in data från standardinmatningen (som normalt är kopplad till tangentbordet). Du får själv välja om du vill att funktionshuvudet ska se ut så här:

```
struct Triangel las_triangel(void)
```

eller så här:

```
void las_triangel(struct Triangel* tp)
```

# Uppgift 7 (5 p)

Vi vill skapa funktionen **min**, som returnerar längden på den kortaste sidan i en triangel, och funktionen **max**, som returnerar längden på den längsta sidan i en triangel. Funktionerna ska ta en triangel (**struct Triangel**), eller en pekare till den, som parameter.

Skriv funktionerna **min** och **max**.

# Uppgift 8 (2 p)

Skriv funktionen **skala** som skalar om en triangel med en faktor, som anges som argument. Om man exempelvis skickar med faktorn **2**, ska alla tre sidorna i triangeln bli dubbelt så långa.

# Uppgift 9 (3 p)

Skriv funktionen **area**, som returnerar arean av en triangel. Beräkna arean **A** med hjälp av följande formel:

$$A=\frac{a}{2}\sqrt{b^2-\left(\frac{a^2+b^2-c^2}{2\mathrm{a}}\right)^2}$$

# Uppgift 10 (4 p)

Skriv en **main**-funktion som har en lokal variabel av triangeltypen, och som gör följande:

- läser in data till variabeln med hjälp av funktionen **las_triangel**
- skriver ut triangeln med **visa_triangel**
- beräknar triangelns area med **area**, och skriver ut den
- frågar efter och läser in en skalfaktor
- applicerar den skalfaktorn på triangeln med funktionen **skala**
- skriver ut den omskalade triangeln med **visa_triangel**
- beräknar den omskalade triangelns area med **area**, och skriver ut den

| |
|---|
| I den här och alla andra uppgifter på tentan gäller:<br>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. |

# Uppgift 11 (3 p)

Skriv funktionen **max_area**, som returnerar arean av den största triangeln i en array. Den ska ta en array med trianglar som argument, samt ett heltal som anger hur många trianglar som finns i arrayen, och den ska alltså returnera arean av den triangel i arrayen som har störst area. Använd funktionen **area** för att beräkna areorna.

# Uppgift 12 (3 p)

Vi vill provköra funktionen **max_area**. Skriv därför en **main**-funktion som anropar **max_area** med lämpliga data, och kontrollerar att den ger det förväntade svaret. Om funktionen ger fel svar, ska ett tydligt felmeddelande skrivas ut. Annars behöver inget skrivas ut.

För en bra testning på riktigt skulle man behöva fler anrop, men här nöjer vi oss med ett enda. Det finns alltså bara ett enda testfall. Försök göra det testfallet så bra som möjligt!

# Uppgift 13 (3 p)

Skriv ett C-program som läser in data om en (ja, bara en) triangel från användaren, med hjälp av funktionen **las_triangel**, och sparar den på en fil. Välj själv om det ska vara en text- eller binärfil. Tala om vilket du valde!

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

*Glöm inte att ange om du valde en textfil eller en binärfil!*

# Uppgift 14 (3 p)

Skriv ett C-program som läser filen från uppgiften ovan, och skriver ut den lagrade triangeln med hjälp av funktionen **visa_triangel**.

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

# Uppgift 15 (1 p)

Vad skriver följande C-program ut? Motivera svaret!

```
#include <stdio.h>

int main(void) {
    double x = 0.2, y = 6.0, z = 1.2;
    printf("%f\n", x * y);
    printf("%f\n", z);
    if (x * y == z)
        printf("Lika, förstås.\n");
    else
        printf("Va? Inte lika!\n");
    return 0;
}
```