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 16 januari 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 23 poäng. För den som följt campuskursen hösten 2013 ger varje i tid inlämnad inlämningsuppgift med deadline en extra poäng. Den som *inte* gått campuskursen hösten 2013 får dessa (tre) extrapoäng ändå.
Resultat och lösningar:	Meddelas via e-post eller på kursens hemsida, http://basen.oru.se/kurser/c/2013-2014-p2/, senast torsdag 6 februari 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.

- 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 + 1 * 1 + 1`

b) `1+1 * 1+1`

c) `1 - 1 / (1 + 1)`

d) `1.0 - 1.0 / (1.0 + 1.0)`

Uppgift 2 (2 p)

Variablerna **a**, **b**, **c** och **d** är av typen **int**. Vilka värden har variablerna efter att följande kod har körts?

```
a = 1; b = 2;
c = a + b;
if (a > b)
    d = 1;
else
    d = 2;
while (a < c) {
    d = d + a;
    a = a + 1;
    if (a > c)
        d = 100;
}
```

Uppgift 3 (4 p)

Skriv ett komplett C-program (med **#include** och allt) som läser in rader, med ett ord på varje rad, ända tills användaren matar in ordet **klar**. Därefter ska programmet tala om hur många gånger användaren skrev vart och ett av orden **luftpistol** och **godis**. Ord med stora bokstäver, som **Godis**, behöver inte räknas. Man kan förutsätta att användaren aldrig skriver några mellanslag, utan bara ett ord per rad.

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**.

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 4 (3 p)

Tärningar finns i många varianter, med olika antal sidor, inte bara sexsidiga. Nu ska vi simulera en tärning. Skriv ett komplett C-program (med **#include** och allt) som först läser in ett heltal som anger antalet sidor på tärningen. Sen ska programmet "slå tärningen" och skriva ut resultatet. Om det till exempel är en 10-sidig tärning, blir resultatet ett heltal mellan 1 och 10 (inklusive gränserna).

Ledtråd:

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));**

Scenario till de följande uppgifterna

Vi vill arbeta med komplexa tal. Ett komplext tal består av en realdel och en imaginärdel, som kan lagras som två vanliga flyttal. Exempelvis kan det komplexa talet **7.4 + 2i** lagras som de två talen **7.4** och **2.0**.

Uppgift 5 (1 p)

Skapa datatypen **struct Komplex**, som ska användas för att representera ett komplext tal.

Uppgift 6 (1 p)

Definiera en variabel som har datatypen **struct Komplex**, och initiera den med det komplexa talet **7.4 + 2i**.

Uppgift 7 (3 p)

Vi vill kunna visa komplexa tal på skärmen. Skriv därför funktionen **visa_komplex**, som skriver ut ett komplext tal. Ett tal som **7.4 + 2i** kan skrivas ut till exempel som **7.40 + 2.00i**.

Om real- eller imaginärdelen är noll, ska den delen inte skrivas ut, så exempelvis **7.4 + 0i** kan skrivas ut till exempel som **7.40**. Talet **0 + 3.14i** kan skrivas ut som **3.14i**. Om både real- och imaginärdelen är noll, ska realdelen skrivas ut.

Uppgift 8 (3 p)

Skriv funktionen **las_komplex**, som läser in ett komplext tal. Funktionen ska skriva ut lämpliga ledtexter på standardutmatningen, och läsa in data från standardinmatningen (som normalt är kopplad till tangentbordet).

Uppgift 9 (2 p)

När man adderar två komplexa tal, adderas realdelarna för sig och imaginärdelarna för sig. Skriv funktionen **komplex_plus**, som tar två komplexa tal som argument, och som returnerar deras summa som ett komplext tal.

Uppgift 10 (3 p)

Absolutbeloppet av ett komplext tal **a + bi** kan i det komplexa talplanet tolkas som avståndet från origo till punkten **(a, b)**, och beräknas med Pythagoras sats: kvadratroten ur summan av kvadraterna på **a** och **b**.

Skriv funktionen **belopp**, som tar ett komplext tal som argument, och returnerar absolutbeloppet som ett flyttal.

Uppgift 11 (3 p)

Skriv en **main**-funktion som läser in två komplexa tal med **las_komplex**, adderar dem med **komplex_plus**, och skriver ut summan med **visa_komplex**. Till sist ska programmet skriva ut summans absolutbelopp.

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.

Uppgift 12 (3 p)

Skriv funktionen **komplex_summering**, som summerar en hel array med komplexa tal. Den ska ta två argument: en array med komplexa tal, och ett heltal som anger antalet tal i arrayen. Den ska returnera summan av alla talen, som också är ett komplext tal.

Uppgift 13 (3 p)

Vi vill provköra funktionen **komplex_summering** från uppgiften ovan. Skriv därför en **main**-funktion som skickar en array med tre komplexa tal till funktionen, med några lämpliga värden, och kontrollerar att resultatet blir det förväntade. Om det resultatet inte blir det förväntade, ska funktionen skriva ut ett tydligt felmeddelande.

Uppgift 14 (4 p)

Skriv ett C-program som läser in komplexa tal med hjälp av funktionen **las_komplex**, och sparar talen på en fil. Inmatningen ska avslutas när användaren matar in talet **0 + 0i**. Det talet ska inte sparas på filen.

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

Uppgift 15 (4 p)

Skriv ett C-program som läser filen från uppgiften ovan och skriver ut summan av de komplexa talen som finns på filen.

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