Teoretisk tentamen, HI1024:TEN1, den 3 juni 2014

Allmänna anvisningar

Då du svarar på en fråga ska det framgå av ditt svar varför du vet att det svar du anger är korrekt, att bara ange det rätta svaret betraktas som ”vagt” eller ”ofullständigt” och ger normalt inte poäng. Vi ska inte testa utantillkunskaper, vi måste testa att ni verkligen har inhämtat materialet och kan svara på frågor som är komponerade på sätt som ni inte sett förut. Instuderingsfrågorna och laborationerna är det bästa sättet att förbereda sig för teoretiska tentan dock på själva tentan förekommer nya kombinationer av begreppen som ni tränat på. Tentamen utgörs av 5 uppgifter med totalt 22p. 11p krävs för E och 10p krävs för Fx.

1. *(1p)* Om man skriver ett stort program med en komplicerad logisk struktur där det uppstår ett stort indenteringsdjup, alltså kod av typen

```
bla bla bla {
        bla bla bla {

                ... Många nivåer ...
                                bla bla bla{
                                        bla bla bla{

                                        }
                                }
        }
}
```

så kan det bli svårt att läsa denna kod. Vad bör man göra i en sådan här situation för att öka läsbarheten?

2. *(3p)* Antag att `a`, `b` och `c` är flyttalsvariabler och att `b` och `c` har värdet `2.0`. Antag också att `summa` är en heltalsvariabel med värdet `23`. Ange vilka värden `a`, `b` och `c` har efter följande kodsnutt (kom ihåg att förklara varför variablerna får de värden som de får):

```
a = summa/10;

if(a<=b) {
  a=a-1.0;
  c=a; a=b; b=c;
  a=a-1.0;
}
```

3. *(6p)* Förklara följande begrepp och ge exempel på användningar:

a) *Lokal variabel* b) *Värdeoperator* c) *Separatkompilerat bibliotek*

4. *(6p)* Betrakta nedanstående struct

```
struct triangel { float a,b,c; }
```

Vi använder den för att lagra representationer av trianglar. Då det gäller trianglar så måste sidorna `a`, `b` och `c` uppfylla något som kallas *triangelolikheten*, vilket innebär att en sida aldrig kan vara längre än summan av längden av de andra två sidorna, i *C* kan vi uttrycka det så här:

```
a<=b+c && b<=a+c && c<=a+b
```

Detta villkor måste alltså vara uppfyllt för att de ingående talen ska vara sidor i en triangel.

Skriv funktionsprototyper för funktionerna `skapa_triangel()`, `triangel_ok()`, `minska_sidor()` så att nedanstående program fungerar:

`skapa_triangel()` används för att skapa en triangel som kallas `t` med sidorna 10.0, 5.0 och 5.0.
loop som kör tills `t` *inte* är en triangel: (`triangel_ok()` används för att kolla)
    `minska_sidor()` används för att minska alla sidor i `t` med 1.

Programmet är skrivet i pseudokod och din uppgift är att skriva funktionsprototyperna utifrån följande förutsättningar:

1. Funktionen `skapa_triangel()` ska ta flyttal som inparametrar och returnera en struct av typen `triangel`.
2. Funktionen `triangel_ok()` ska ta en triangel som inparameter och returnera ett heltal som är 1 om det är en triangel som skickas in och annars 0.
3. Funktionen `minska_sidor()` ska användas för att kunna påverka en triangel på så sätt att alla sidor minskas med 1.

5. *(8p)* Vi arbetar vidare med trianglar. Betrakta nedanstående program. Det är tänkt att slumpa ut trianglar av varierande dimensioner och sortera dem på area respektive omkrets. Men programmet är ofullständigt, det saknas anrop till fyra funktioner på fem olika rader i programmet. Komplettera programmet så att det fungerar som tänkt. (Du ska endast ändra 5 rader i nedanstående program, den tredje och femte raden du ändrar ska vara likadana eftersom de ska fungerar på samma sätt: bara skriva ut trianglarna.)

```
#include <stdio.h>
#include "trianglar.h" //Biblioteket trianglar är givet på nästa sida

main()
{
  struct triangel t[10];  int i;
  srand(time(0));

  ANROP TILL slumpa_trianglar() SAKNAS
  ANROP TILL sortera() SAKNAS
  printf("Sorterade på omkrets:\n\n");
  for(i=0;i<10;i++)
  ANROP TILL skriv_triangel() SAKNAS (samma anrop som nedan)
  ANROP TILL sortera() SAKNAS
  printf("\nSorterade på area:\n\n");
  for(i=0;i<10;i++)
  ANROP TILL skriv_triangel() SAKNAS (samma anrop som ovan)
}
```

En testkörning:

Sorterade på omkrets:

5.20 1.90 3.50 o: 10.60 a: 2.55
6.00 5.20 2.00 o: 13.20 a: 7.14
6.10 0.60 6.60 o: 13.30 a: 1.49
8.70 4.50 4.50 o: 17.70 a: 7.09
3.80 8.60 5.40 o: 17.80 a: 9.76
7.70 3.60 7.40 o: 18.70 a: 18.60
3.90 9.40 6.30 o: 19.60 a: 12.72
3.50 7.30 9.00 o: 19.80 a: 17.22
2.20 9.40 9.80 o: 21.40 a: 14.59
7.10 7.20 7.40 o: 21.70 a: 32.01

Sorterade på area:

6.10 0.60 6.60 o: 13.30 a: 1.49
5.20 1.90 3.50 o: 10.60 a: 2.55
8.70 4.50 4.50 o: 17.70 a: 7.09
6.00 5.20 2.00 o: 13.20 a: 7.14
3.80 8.60 5.40 o: 17.80 a: 9.76
3.90 9.40 6.30 o: 19.60 a: 12.72
2.20 9.40 9.80 o: 21.40 a: 14.59
3.50 7.30 9.00 o: 19.80 a: 17.22
7.70 3.60 7.40 o: 18.70 a: 18.60
7.10 7.20 7.40 o: 21.70 a: 32.01

Biblioteket trianglar:

```
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <stdlib.h>

#define SORT_OMKR 0
#define SORT_AREA 1

struct triangel {
  double a,b,c;
};

double slumptal();

double area (struct triangel t);

double omkrets(struct triangel t);

void skriv_triangel(struct triangel t);

void slumpa_trianglar(struct triangel t[], int antal);

void byt(struct triangel *t1, struct triangel *t2);

void sortera(struct triangel t[], int antal, int sortera_efter);

//(Om sortera_efter==SORT_AREA) så sker sorteringen efter area och
//motsvarande gäller med omkrets.
```