

Fråga 1

Vilken uppgift har main-funktionen i ett C-program ?

Ett C-program består av en eller flera funktioner, åtminstone funktionen main. main är den funktion som anropas automatiskt när programmet exekveras.

Fråga 2

Vad är en funktions prototyp och vilken uppgift har den?

Prototyp är funktionshuvudet, möjligtvis med undantag för de formella parametrarnas namn. Returtyp, funktionsnamn och parametrarnas typer (till antal och ordning) utgör prototypen.

Funktionsprototypen förser kompilatorn med information om funktionen utan att funktionsdefinitionen finns med. Alla identifierare som finns i källkoden måste kunna identifieras av kompilatorn, dvs. vara kända till typ och därmed användningsmöjlighet.

Fråga 3

```
#include <stdio.h>
```

Vad händer när ovanstående rad (eller någon liknande) finns i ett C-program?

Varför behövs den?

Alla rader som inleds med # är kommandon till preprocessor, vilket är ett program som "förbehandlar" källkodsfilen före kompileringen.

#include anger att den efterföljande informationen är en fil som skall "klistras in" precis som den är och precis på denna plats. Filer med extensionen .h innehåller funktionsprototyper, vilket gör det möjligt att använda storheter som inte finns definierade i den aktuella källkodsfilen.

Fråga 4

Beskriv skillnaden mellan formella och aktuella parametrar.

Formella parametrar är de parametrar som deklarerats i funktionshuvudet i funktionsdefinitionen. Dessa är "lokala" namn som endast existerar under den tid som koden i funktionen utförs, när man lämnar funktionen försvinner dessa identifierare. De argument (variabler, uttryck) som anges vid funktionsanropet är de **aktuella parametrarna**. Dessa ska till antal, typ och ordning motsvara de formella parametrarna.

Fråga 5

Fyll i följande tabell, visa med parenteser

Deklarationer och initialiseringar		
<pre>int i=1, j=2, k=3; char c = 'B';</pre>		
Uttryck	Evalueringsordning	Värde
<code>i==j</code>	falskt	0
<code>i=2</code>	tilldelning och värdet sant	2
<code>i==2</code>	falskt	0
<code>i!=2</code>	sant	1
<code>i+j+k == -2*-k</code>	$((i+j)+k) == ((-2)*(-k))$ 6 == 6 sant	1
<code>i && j && k</code>	$((i) \&\& (j)) \&\& (k)$ $((1) \&\& (2)) \&\& (3)$ $(s \ \&\& \ s) \ \&\& \ s$ sant && sant sant	1
<code>c-1=='A' c+1=='Z'</code>	$((c-1)=='A') ((c+1)=='Z')$ $((66-1) == 65) (67==90)$ sant falskt sant	1

Fråga 6

Du vill summera de udda heltalen mellan 1 och n (n är udda),
dvs. $1 + 3 + 5 + 7 + \dots + n$

Utgå från följande deklARATIONER

```
int sum,i;
int n; /* n får värde på något sätt */
```

och gör summationen med

- en for-loop

```
sum = 0;
for (i=1; i<=n; i = i + 2)
    sum = sum + i;
printf("For-loop ger sumUdda(1..%i) = %i\n\n",n,sum);
```

- en while-loop

```
sum = 0;
i = 1;
while (i<=n)
{
    sum = sum + i;
    i = i + 2;
}
printf("While-loop ger sumUdda(1..%i) = %i\n\n",n,sum);
```

- en do-while loop

```
sum = 0;
i = 1;
do
{
    sum = sum + i;
    i = i + 2;
}while (i<=n);
printf("doWhile-loop ger sumUdda(1..%i) = %i\n\n",n,sum);
```

Körning :

For-loop ger sumUdda(1..7) = 16

While-loop ger sumUdda(1..7) = 16

doWhile-loop ger sumUdda(1..7) = 16

Fråga 7

Vilka värden har `v1` och `v2` efter funktionsanropet som är markerat?

Förklara vad som händer.

```
a) v1[0] = 11; v1[1] = 22; v1[2] = 33; v1[3] = 44; v1[4] = 55;
   v2[0] = 99; v2[1] = 99; v2[2] = 99; v2[3] = 99; v2[4] = 99;
   fixaElement(v1);
   skrivLista("fixaElement:", v1, 5);
```

Körning :

```
Test av fixaElement:
11 22 77 44 88
```

Kommentar : adressen i fältnamnet `v1` kopieras till den formella parametern `lista`, `lista` och `v1` refererar nu till samma minnesutrymme. I funktionen ändras två av värdena, vilket alltså sker direkt i det minnesutrymme som den aktuella parametern `v1` refererar till.

```
b) v1[0] = 11; v1[1] = 22; v1[2] = 33; v1[3] = 44; v1[4] = 55;
   v2[0] = 99; v2[1] = 99; v2[2] = 99; v2[3] = 99; v2[4] = 99;
   fixaReferens(v1, v2);
   skrivLista("flyttaReferens:", v1, 5);
```

Körning :

```
Test av fixaReferens:
11 22 33 44 55
```

Kommentar : adressen i fältnamnet `v1` kopieras till den formella parametern `lista1` och adressen i fältnamnet `v2` kopieras till den formella parametern `lista2`. I funktionen ändras värdet på `lista1` med en tilldelning till samma värde som `lista2`. Dessa refererar nu till samma utrymme och dessutom till samma utrymme som `v2`. Eftersom ev. förändringar i de formella parametrarnas värden inte påverkar motsvarande aktuella parametrar (i det här fallet är värdena adresser) så har ingenting hänt med den aktuella parametern `v1`.

```
c) v1[0] = 11; v1[1] = 22; v1[2] = 33; v1[3] = 44; v1[4] = 55;
   v2[0] = 99; v2[1] = 99; v2[2] = 99; v2[3] = 99; v2[4] = 99;
   kopieraArray(v1, v2, 5);
   skrivLista("kopieraArray:", v1, 5);
```

Körning :

```
Test av kopieraArray:
99 99 99 99 99
```

Kommentar : adressen i fältnamnet `v1` kopieras till den formella parametern `lista1` och adressen i fältnamnet `v2` kopieras till den formella parametern `lista2`. I funktionen kopieras de enskilda heltalsvärdena ett i taget från en plats i minnet, med basadress `&lista1`, till motsvarande plats utgående från basadressen `&lista2`. De formella parametrarna har inte ändrat värde (`lista1` och `lista2`'s värden är ju adresser). `v1` och `v2`'s värden är oförändrade, medan

d) `v1[0] = 11; v1[1] = 22; v1[2] = 33; v1[3] = 44; v1[4] = 55;`
`v2[0] = 99; v2[1] = 99; v2[2] = 99; v2[3] = 99; v2[4] = 99;`
`v1 = returneraReferens (v2);`
`skrivLista ("returneraReferens v1:", v1,5);`
`skrivLista ("returneraReferens v2:", v2,5);`

Körning :

```
Test av returneraReferens v1:
99 9876 99 99 99
Test av returneraReferens v2:
99 9876 99 99 99
```

Kommentar : adressen i fältnamnet `v2` kopieras till den formella parametern `lista`, dessa refererar nu till samma utrymme.

I funktionen sätts värdet på `lista[1]` till `9876` med en tilldelning.

Eftersom ev. förändringar i de formella parametrarnas värden inte påverkar motsvarande aktuella parametrar (i det här fallet är värdena adresser) så har ingenting hänt med den aktuella parametern `v1`.