Några tips kring hur man kan hantera funktionerna `scanf()`, `gets()` och `flush()`

För det första är dess funktioner potientiellt skadliga, speciellt `gets()`, men detta är en introduktionskurs och att läsa korta inmatningar är inte några bekymmer. För den mer intresserade finns en bra webbsida som beskriver problemen med funktionerna, googla på "*Things to avoid in C/C++*".

Dessa tips är till för att ni ska kunna lösa den sista laborationen och övningen till den sista laborationen bättre.

När man matar in saker till ett program skrivet i *C* finns det något som heter *inmatningsbufferten*. Den lagrar tecken i respons till tangentryckningar och om man skriver in talet 5, följt av returtangenten har bufferten följande innehåll:

```
| 5 | \n |
```

När man sedan läser med satsen `scanf("%d", &x)`, så läggs talet 5 in i variabeln `x`. Vi har också sett hur man kan läsa in strängar med `scanf()` eller `gets()`, om vi skriver in strängen *"Johnny"* och trycker på retur ser inmatningsbufferten ut så här:

```
| J | o | h | n | n | y | \n |
```

och anropet `gets(namn)` får *"Johnny"* att lagras i lagras i strängen `namn`, förutsatt förstås att namn är deklarerad som `char[]`. Som vi även känner till kan man även läsa in strängen med satsen `scanf("%s", namn)`. (Och här ska vi alltså inte ha en adressoperator (`&`) framför `namn`.)

Så långt är allt frid och fröjd, det blir dock bekymmer när vi ska kombinera inmatning av tal och text. Om vi vill rent spontant skriver

```
int x;
char namn[100];

printf("Mata in x: ");
scanf("%d", &x);

printf("Mata in namn: ");
gets(namn);

printf("Tal: %d\n", x);
printf("Namn: %s\n", namn);
```

och tänker oss att vi först matar in ett heltal, kanske 5, i variabeln `x` och sedan en sträng, kanske *"Johnny"*, i variabeln `namn` så kommer vi att bli förvånade över att finna att programmet faktiskt inte fungerar. Det som verkar hända om man provkör ovanstående program är att programmet verkar hoppa över satsen `gets(namn)`, vi får inte ens en chans att mata in `namn`. Förklaringen är enkel, returtangenttryckningen efter inmatningen av heltalet 5 ligger faktiskt kvar i inmatningsbufferten. Funktionen `scanf()` tar inte bort `\n` som kommer efter 5. Det betyder att `gets()`, som läser från inmatningsbufferten tills det kommer ett `\n` uppfattar att inläsningen redan

är fullbordad. Lösningen är att tömma bufferten och det göra man enklast genom kommandot `fflush(stdin);`. Vårt rättade program skulle då få utseendet:

```
int x;
char namn[100];

printf("Mata in x: ");
scanf("%d", &x); fflush(stdin);

printf("Mata in namn: ");
gets(namn);

printf("Tal: %d\n", x);
printf("Namn: %s\n", namn);
```

Detta program bör fungera som det är tänkt. Vi sammanfattar detta till en allmän regel: så fort vi använt `scanf()`, gör `fflush(stdin)`. Provkör dessa båda program.

Nu är inte några av dessa funktioner så bra alls (se webblänken ovan), men de fungerar i en grundkurs som vi går igenom här. I framtiden kommer ni att troligen programmera i andra programmeringsspråk och/eller ha bättre hantering av in och utmatning.