

Mera om printf

- **printf** har ett stort antal **flaggor** och **modifierare** som kan användas för att få exakt den utskrift som önskas för olika variabler
 - Antal decimaler, “padding”, etc
 - Se tabellerna 16.1 – 16.4 i kursboken för en komplett förteckning
- **printf** kan skriva ut värdet på **flera** variabler med ett och samma anrop

Mera om printf, exempel

```
#include <stdio.h>

int main(void)
{
    int var1 = 23;
    double var2 = 45.237;
    char var3 = 'z';

    // Let's do some fancy formatting
    printf("var1 with 5 'padding spaces': %5d\n", var1);
    printf("var2 with 2 decimals: %.2f\n", var2);

    // Let's output 3 values with one printf
    printf("var1 is %d, var2 is %f, var3 is %c\n",
           var1, var2, var3);

    return 0;
}
```

Inmatning av data - **scanf**

- **scanf** kan ses som en direkt motsvarighet till **printf**: **printf** skriver ut information, **scanf** läser in information (via tangentbordet)
- Den inmatade information lagras i en variabel
- Viktigt att ange rätt datatyp vid inmatning av data
 - I stort sett samma syntax som för **printf**: t.ex. motsvarar **%d** ett heltal.
 - **Undantag**: Vid inläsning av en **double** måste syntaxen **%lf** användas för att mata in ett “långt flyttal”; endast **%f** läser in en **float**

scanf, exempel

```
#include <stdio.h>

int main(void)
{
    int num1, num2;
    double num3;

    printf("Please input an integer! ");
    // Note: & before the variable name!
    scanf("%d", &num1);
    printf("You entered %d\n", num1);

    printf("Please input an integer and a double! ");
    scanf("%d%lf", &num2, &num3);
    // Note: You can also use %lf in printf
    printf("You entered %d and %lf\n", num2, num3);

    return 0;
}
```

Inmatning av data - **scanf**

- &-tecknet framför variabelnamnen i **scanf** är inte ett tryckfel
 - Denna syntax används för att komma åt *minnesadressen* till en variabel – mera om detta i samband med avsnittet om minneshantering
- Vid inmatning av flera värden kan dessa separeras med mellanslag, tabulator eller radbyte
- Tills vidare gör vi det överoptimistiska antagandet att användaren alltid matar in data av rätt typ

scanf och char-inläsning

- **scanf** läser in data från en inputbuffer
- Det radbytestecken, mellanslag, etc. som avslutat inmatningen lämnas kvar i buffern efter inläsningen
- Problem vid upprepad inmatning:
scanf("%c") kommer att läsa in detta 'avslutningstecken' och användaren får aldrig en chans att mata in någon ny data
- Kan instruera **scanf** att ignorera inledande whitespace genom att lägga in ett extra mellanslag i 'inputsträngen':
scanf(" %c", &charvar);

scanf och char-inläsning

```
char c1, c2;

// First try - won't work
printf("Please input first char!\n");
scanf("%c", &c1); // Terminating character is left in buffer
printf("Please input second char!\n");
scanf("%c", &c2); // Will read the terminating char from buffer

printf("You entered %c and %c\n", c1, c2);

// Second try - skip whitespace before actual data
printf("Please input first char!\n");
scanf("%c", &c1); // Terminating character is left in buffer
printf("Please input second char!\n");
scanf(" %c", &c2); // Ignore terminating character

printf("You entered %c and %c\n", c1, c2);
```