Variabler

- Ett av de viktigaste och mest grundläggande begreppen inom programmering
- Kan ses som en namngiven plats i datorns minne där data kan lagras
- Innehållet kan modifieras under programkörningen
- Innehållet kan t.ex. skrivas ut genom en **`printf`** -instruktion
- Olika *typer* av variabler – i våra första exempel använder vi variabler som innehåller *heltal*

Deklaration av variabler

- En variabel måste *deklareras* innan den kan användas i ett program
- Berättar för kompilatorn vad variabeln skall heta samt vilken typ av data den skall innehålla
- Med hjäp av *tilldelningsoperatorn* **=** kan en variabel tilldelas ett värde
  - Kan göras både i samband med deklarationen och senare under programkörningen
- En variabel kan användas endast i det kodblock där den är deklarerad

Variabler, exempel

```
#include <stdio.h>

int main(void)
{
    /* Declare a variable of type integer, name it sum and
       initialize it to zero */
    int sum = 0;

    /* Change the value of sum, just to show that we can */
    sum = 10 + 20;

    /* Output the value of sum using the printf function.
       The format specifier %d tells printf to output the
       value of an integer (digit) variable as part of the string.
       This variable is then sent as the second parameter
       to printf */
    printf("The sum of 10 and 20 is %d\n", sum);

    return 0;
}
```

Namngivning av variabler

- Ett variabelnamn måste inledas med en bokstav eller “underscore”-tecknet _
- Det inledande tecknet kan följas av valfri följd bokstäver, siffror och underscores
- Specialtecken är inte tillåtna
- *Reserverade ord* är inte tillåtna – en variabel kan t.ex. inte heta **`int`** eller **`return`**
- Se kapitel 1.2.2 i Appendix A i kursboken för en lista över reserverade ord i C

Datatyper för heltal

- Datatypen **int** används för att representera heltal
- En **`int`** är garanterad att inrymma minst 32 *bitar* (= 4 bytes) information
  => heltalsvärden inom intervallet (cirka) ±2 miljarder
- En **`unsigned int`** representerar endast positiva värden => heltalsvärden mellan 0 och ca 4 miljarder
- I C ges inga garantier för exakt hur många bytes en viss datatyp motsvarar – operatorn **`sizeof`** kan användas för att kontrollera den exakta storleken.

Datatyper för decimaltal

- Decimaltal representeras i C med datatyperna **`float`** och **`double`** – kallas även för "flyttal"
  - **`float`** motsvarar typiskt 32 bitar
  - **`double`** motsvarar typiskt 64 bitar
- **Viktigt:** Decimaltal kan i allmänhet inte representeras exakt i en dators minne
- Ju flera bitar som finns tillgängliga, desto noggrannare representation
- **Rekommendation**: Använd **`float`** endast i undantagsfall
- **Rekommendation**: Föredra heltal framför flyttal

Flyttalsvariabler, exempel

```
#include <stdio.h>

int main(void)
{
    /* Declare a variable of type double, name it sum and
       initialize it to zero */
    double sum = 0;

    /* Change the value of sum */
    sum = 10.5 + 20.7;

    /* Output the value of sum using the printf function.
       The format specifier %f tells printf to output the
       value of a float variable. */
    printf("The sum of 10.5 and 20.7 is %f\n", sum);

    return 0;
}
```

Datatyper för tecken

- Datatypen **`char`** kan användas för att lagra bokstäver / tecken
- Varje tecken motsvaras egentligen av siffervärden i *ASCII-tabellen (http://www.asciitable.com/)*

  => En **`char`** kan ha värden mellan 0 – 255 (8 bitar, en byte).
- Finns ett antal användbara *specialtecken* som radbyte, tabulator, etc.
  - Se Appendix A, tabell A3 i kursboken för ytterligare specialtecken
- Oftast krävs hantering av många tecken samtidigt
  => Behov av *strängar*
  - Behandlas senare under kursen

Teckenvariabler, exempel

```
#include <stdio.h>

int main(void)
{
    /* Declare some variables of type char, name them and
       initialize them */
    char aVariable = 'a'; // Note the single
                          // quotation marks

    char newLine = '\n';  // Special character,
                          // '\n' counts as one char

    /* Output the values using the printf function.
       The format specifier %c tells printf to output the
       values as characters. */
    printf("This is a character: %c\n", aVariable);
    printf("This is another character: %c\n", newLine);

    return 0;
}
```

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
  - Viktigt att antal och ordningsföljd för 'format specifiers' matchar variabelparametrarna

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;
}
```

“Type specifiers”

- “Tilläggsdirektiv” som kan användas för att modifiera variablers storlek / lagringskapacitet
- **`unsigned`** tillåter endast positiva värden
- **`long`** används för att utöka kapaciteten för variabler
- **`short`** används för att minska kapaciteten (och därmed spara minnesutrymme)
- Om du endast anger datatypen som **`unsigned`**, **`long`**, **`short`**, osv. antar kompilatorn att du vill deklarera en **`int`** med dessa egenskaper.

Type specifiers, exempel

```
#include <stdio.h>

int main(void)
{
    /* Declare a variable of type short int */
    short int small_number = 32;

    /* Change the value of the variable to something big */
    small_number = 1500000000;

    /* Output the value */
    printf("Value is %d\n", small_number);

    return 0;
}
```

sizeof, exempel

```
// Can use sizeof to check the size in bytes
// of data types or variables
// ANSI C99 defines a special output format for sizeof: %zu
// However, this does not work with all compilers

printf("Size of char is %d\n", sizeof(char));
char char_var;
printf("Size of char_var is %d\n", sizeof(char_var));

printf("Size of short is %d\n", sizeof(short));
printf("Size of int is %d\n", sizeof(int));
printf("Size of unsigned int is %d\n", sizeof(unsigned int));
printf("Size of long is %d\n", sizeof(long));
printf("Size of float is %d\n", sizeof(float));
printf("Size of double is %d\n", sizeof(double));
printf("Size of long double is %d\n", sizeof(long double));
```