Aritmetiska uttryck

- Vi har redan sett exempel på några enkla aritmetiska uttryck: addition(**+**) och subtraktion(**-**)
- Övriga aritmetiska operatorer:
  ***** (multiplikation)
  / (division)
  % (“modulus”, “rest vid heltalsdivision”)
- Viktiga begrepp vid beräkning av aritmetiska uttryck: *operatorprecedens* och *reglerna för heltalsaritmetik*

Enkla aritmetiska uttryck, exempel

```
int a=22, b=5; /* Note that several variables can be declared
                  simultaneously! */
int c=a*b;

printf("c equals %d\n", c);
printf("a * b equals %d\n", a*b); /* We don't actually need
                                     to use an intermediate
                                     variable */

printf("a / b equals %d\n", a/b);
printf("a % b equals %d\n", a%b);

/* What do you think the output of the last two statements
will be? */
```

Operatorprecedens

- Ett uttryck som innehåller många operatorer beräknas i enlighet med de olika operatorernas *precedens* eller prioritet
- Samma regler gäller som då ett uttryck beräknas för hand
- Multiplikation och division har t.ex. högre precedens än addition och multiplikation
  - 8*2+5*4 = 16+20 = 36
- Med hjälp av parenteser kan precedensreglerna ändras
  - 8*(2+5)*4 = 8*7*4 = 224

Operatorprecedens

- Rekommendation: Använd parenteser även då detta i princip inte är nödvändigt
  - (8*2)+(5*4)
  - Mer lättläst kod
  - Mindre risk för buggar
- Se tabell A.5 i kursboken för en komplett lista över operatorer och deras respektive prioritet.

Heltals- och flyttalsaritmetik, exempel

```
int a=25;
int b=2;

/*
   Dividing and then multiplying with the same factor
   should result in the original value, right?;
   Let's try it out!
*/

printf("25 / 2 * 2 = %d\n", a/b*b);

/* What is the actual result? Why? */
```

Heltals- och flyttalsaritmetik, exempel

- Ett uttryck med två heltalsoperander resulterar I ett heltalsresultat
- En eventuell heltalsdel “slängs bort”
- Måste använda flyttal för att behålla decimaldelen

```
int a1 = 25, b1 = 2;
double a2 = 25.0, b2 = 2.0;
/* What are the results of these calculations? */
printf("25 / 2 * 2 = %f\n", a1/b1*b1);
printf("25 / 2 * 2 = %f\n", a2/b2*b2);
printf("25 / 2 * 2 = %f\n", a1/b2*b2);
printf("25 / 2 * 2 = %d\n", a1/b2*b2);
```

“Type cast”-operatorn

- Ett annat sätt att lösa problemet med försvinnande decimaldelar: Omvandla en datatyp till en annan med *type cast*-operatorn ()

```
int a1 = 25, b1 = 2;

printf("25 / 2 * 2 = %f\n", (double)a1/b1*b1);
```

- I exemplet ovan omvandlas heltalet i **`a1`** till en **`double`** (med värdet 25.000...)
  => resultatet av hela beräkningen blir en **`double`**
  (Vad säger dig detta om type cast-operatorns *precedens*/*prioritet*?)

Tilldelningsoperatorn

- Som vi redan har sett kan en variabel tilldelas ett värde med hjälp av *tilldelningsoperatorn* =
  ```
  int a = 5;
  ```
- En variabels värde kan ändras i ett senare skede av programmet:
  ```
  a = 237;
  ```
- En variabels värde kan bero av *en annan variabels värde*:
  ```
  int b = 12;
  a = 182 * b;
  ```
- En variabels nya värde kan bero av *samma variabels tidigare värde*:
  ```
  a = a + b / 2;
  ```

Tilldelningsoperatorn

- Alternativ syntax för att öka eller minska värdet av en heltalsvariabel med ett:

```
a++; // motsvarar a = a + 1;
a--; // motsvarar a = a – 1;
```

- Alternativ syntax för att ändra en variabels värde utgående från dess tidigare värde:

```
a -= 10;     // motsvarar a = a – 10;
a *= b;      // motsvarar a = a * b;
a += b / 2;  // motsvarar a = a + b / 2;
```

Tilldelningsoperatorn

- Vid deklaration och initialisering av flera variabler samtidigt måste alla variabler initialiseras med ett eget värde

```
int a, b; // values of a and b are undefined

int a=0, b; // value of b is undefine

int a, b=0; // value of a is undefined

int a=0, b=0; // both a and b are initialized
```

- Rekommendation: **Intialisera alltid alla variabler**!