Föreläsning 2

- Variabler, tilldelning och kodblock{}
- if-satsen
- Logiska operatorer
- Andra operatorer
- Att programmera

Variabler

- Det är i variabler som all data (information) lagras.
- Genom att ändra värde på variabler kan samma programmeringssats utföra olika saker / få olika resultat: s = v*t;

Variabel-varierbar (eng. variable)

Deklarera och tilldela variabler

- Beroende av vilken sorts data som skall lagras använder man variabler av olika datatyper (sorter).
- Innan en variabel kan användas måste man deklarera vilken typ (sort) den är. Ex: int i;
- Sedan måste man tilldela variabeln ett värde:
  i = 3;
- Detta kan göras på samma rad:
  int i = 3;
- Variabeln kan senare tilldelas ett nytt värde. Observera dock att den inte får deklareras igen!
- = -betyder tilldelning ***inte*** lika med. Vad betyder:
  i = i +2;

Exempel på datatyper

Typ	**min/max**	**Egenskaper**
int	-2147483648/2147483647 (garanterat -32768/32767)	32-bit +/- heltal
long	-2147483648/2147483647	32-bit +/- heltal
float	3.4E-38/1.7E38 (7 signifikanta siffror)	32-bit +/-decimaltal
double	1.7E-308/3.4E308 (15 signifikanta siffror)	64-bit +/-decimaltal

Exempel på deklarationer och tilldelningar

```
#include <stdio.h>

int main(void)
{
	int a,b,c=7;
	float antal,pris=3.70,laengd;
	a=34; b=32767;
	antal=-123.978;
	pris=89.00;
	laengd=56;
	c=-38;
	laengd=laengd+1;
}
//Från boken
```

Namngivning

- Välj namn med omsorg. Namnet ska berätta vad variabeln lagrar och göra koden läsbar.
- Variabler brukar ges små bokstäver och konstanter stora
- Om namnet innehåller flera ord: antalPersoner eller antal_personer
- Regler:
  - Engelska (stora eller små) bokstäver får användas (inte å, ä, ö).
  - Siffror 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 får användas.
  - Underscore _ får användas
  - **Ett namn får inte inledas med en siffra.**
  - reserverade (upptagna) ord får ej användas

Kodblock {}

- {} – klumpar ihop flera satser till ett kodblock som utifrån ses som en sats.
- Ex:

```
{
        int i;
        i =0;
        i=i+1;
}
```

Deklarationers giltighet

En deklaration gäller nedanför där den är gjord inom sitt kod-block { }.

```
{
    i = 7; //fel i är inte deklarerad
    int i; //nu är i deklarerad
    i=4; //går bra programmet vet att i är ett heltal
}
i=i+1; //fel i är inte definierad utanför kodblocket{}
```

- Den gäller också innanför kod-block efter deklarationen:

```
int i = 0;
{
    i++; //OK!
}
i = i * i;//OK
```

Villkor

- Om du ger mig hundra kronor så får du en biobiljett.
- Om du hjälper mig med städningen så följer jag med dig på hockey, annars får du gå själv.
- Om klockan är mer än tio så går jag och lägger mig, annars dricker jag kaffe.

if-satsen

- if(*logiskt_uttryck*)
  sats

- int tid = 21;
  if(tid > 22)
  printf(”Gå och lägg dig”);

if else

- if(*logiskt_uttryck*)
  sats1
  else
  sats2
- int tid = 21;
  if(tid > 22)
  printf(”Gå och lägg dig”);
  else
  printf(”Drick kaffe!”);

if { }

- Med hjälp av klammrar kan vi göra flera saker i en if-sats.
- if (tid > 22)

```
{
        printf(”Borsta tänderna!”);
         printf(”Gå och lägg dig!”);
}
else
{
         printf(”Koka kaffe”);
         printf(”Drick kaffe”);
}
```

Flera if

- Vi kan ha en if sats i en annan if sats – nästlat
- Ex:

```
if(0<x){
        if(x<10) printf(”0<x<10”);
}
```

- En särskild form av detta är if else if:

```
if(x<0)
{
        sats1
}
else if(x<10)
{
        sats2
}
else
{
        sats3
}
```

Logik och programmering

- Ett logiskt värde är antingen sant eller falskt
- När vi jämför tal är resultatet antingen sant eller falskt:

  3 < 7 sant
  3 > 7 falskt
  3 == 7 falskt
- Resultatet av sådana jämförelser är mycket användbara i programmering. Vi kan säga åt datorn att göra något om resultatet är sant och något annat om det är falskt.
- I C är resultatet av en sådan jämförelse av typen int och sant representeras av 1 och falskt av 0.
  (*dock tolkas alla värden utom 0 som sanna*)

Relationsoperatorer

- == lika med

  3==4 är falskt dvs resultatet blir 0
- != skilt från

  3!=4 är sant dvs resultatet blir 1
- > större än
- >= större än eller lika med
- < mindre än
- <= mindre än eller lika med

Logiska operatorer

- Exempel på logiska uttryck:

  icke (sant) = falskt

  sant och sant = sant

  sant eller falskt = sant

- Logiska operatorer tar en eller två logiska värden och resultatet blir ett logiskt värde
- *I C motsvaras då sant och falskt av 1 och 0*

Logiska operatorer i C

- Icke - !
  !(1) blir 0
  !(0) blir 1
- och - &&
  1&&1 blir 1
  1&&0 blir 0
  0&&1 blir 0
  0&&0 blir 0
- Eller - ||
  1||1 blir 1
  1||0 blir 1
  0||1 blir 1
  0||0 blir 0

Ett exempel

```
if (x<0 || 10<x)
{
    printf(”x är inte mellan 0 och 10”);
}
```

Operatorer

- En operator tar ett eller två data och producerar ett svar.
- Typexemplet är +. Den tar t.ex två heltal och producerar ett annat heltal nämligen summan: 3 + 5 blir 8

Vanliga matematiska operatorer

- *, /, +, - fungerar som på era miniräknare.
  Fungerar både för heltal och decimaltal (flyttal).
- Heltalsdivision är lite speciell:
  5 / 2 blir 2
- % - modulus, dvs resten vid heltalsdivision:
  5 % 2 blir 1
- ++ och – – är två speciella operatorer:
  a++; motsvarar a=a+1; dvs a ökar med 1
  a – –; motsvarar a=a-1;

  OBS! 4++ ger kompileringsfel!

Konverteringsoperatorn

- (typ) – denna operator konverterar om möjligt talet till höger till datatypen typ.
- Ex: int i = (int) 3.45;
- Konverteringen sker dock automatiskt vid behov:

  int i = 3.45;//ger samma resultat

Konvertering i uttryck

- float f = 3 / 2; //f blir 1.0
- float f = 3.0 / 2; //f blir 1.5

- int i1 = 3;
  int i2 =2;
  float f = i1 / i2; //f blir 1.0
  float f = (float) i1 / i2; //f blir 1.5

Prioritering

- I ett uttryck med flera operatorer måste de göras i en viss ordning. Vilken bestäms av operatorernas prioritet
- Du kan precis som i matematiken ändra prioritetsordningen med hjälp av parenteser. Dessa beräknas först.
- Det är bättre att ha en parentes för mycket än en för litet. Sätt alltid ut parenteser om du är osäker!

Prioritetsordning

1. ()
2. ++ – –
3. (typ)
4. * / %
5. + –
6. =

Om två operatorer har samma prioritet utföres de från vänster till höger

Använd alltid parenteser om du är osäker!

ex: (float)a/b+c/(d+e)

Att programmera

- Skriv en rad i taget
- Kompilera och testkör mellan varje rad
- Om du får kompileringsfel bör felet vara på senaste raden
- Rätta bara första kompileringsfelet – övriga kan vara följdfel som försvinner av sig själv
- Välj beskrivande variabelnamn
- Indentera (tabba) och lämna blankrader så att koden blir lättläst
- Försök läsa kod så som kompilatorn gör:
  Läs första raden – vad händer
  Läs nästa rad vad händer – osv
- Använd extra printf – satser för att följa koden

Ett exempel

- I den här uppgiften skall du skriva ett program som räknar ut priset för CD-R skivor. Grundpriset är 9,90 kr, men om man köper flera får man rabatt:

  fler än 10 st ger 5% rabatt
  fler än 50 st ger 10% rabatt

  Programmet skall börja med att fråga användaren hur många skivor han vill köpa för att sedan svara vad totala priset blir.

Ett exempel

```
#include <stdio.h>

int main(void)
{
        printf("Valkommen!\nHur manga skivor vill du kopa?");

}
```

Ett exempel

```
#include <stdio.h>

int main(void)
{
	int antal;

	printf("Valkommen!\nHur manga skivor vill du kopa?");
	scanf("%d",&antal);

}
```

Ett exempel

```
#include <stdio.h>

int main(void)
{
        int antal;

        printf("Valkommen!\nHur manga skivor vill du kopa?");
        scanf("%d",&antal);

        printf(”du vill köpa %d",antal);

}
```

Ett exempel

```
#include <stdio.h>

int main(void)
{
        int antal;
        float pris;

        printf("Valkommen!\nHur manga skivor vill du kopa?");
        scanf("%d",&antal);

        pris=antal*9.9;
         printf("Priset blir: %.0f",pris);

}
```

Ett exempel

```
#include <stdio.h>

int main(void)
{
        int antal;
        float pris;

        printf("Valkommen!\nHur manga skivor vill du kopa?");
        scanf("%d",&antal);

        if(antal<11)
        {
           pris=antal*9.9;
        }
        else
        {
           pris=antal*9.9*0.9;
        }

         printf("Priset blir: %.0f",pris);

}
```

Ett exempel

```
#include <stdio.h>

int main(void)
{
   int antal;
        float pris;

        printf("Valkommen!\nHur manga skivor vill du kopa?");
        scanf("%d",&antal);

        if(antal<11)
        {
           pris=antal*9.9;
        }
        else if(antal<51)
        {
           pris=antal*9.9*0.95;
        }
        else
        {
           pris=antal*9.9*0.9;
        }

        printf("Priset blir: %.0f",pris);

}
```

Buggar

Programmet nedan innehåller fem fel. Försök finna dessa:

```
void main(void)
{
	int x;
	float y, sum;

	printf("mata in ett heltal: ");
	scanf("%d", x);
	printf("mata in ett flyttal: ");
	scanf("%d", &y);

	sum=x+y;

	printf("Summan är ", sum);
	if(x<0);
		printf("X är negativt");
}
```

Fel

- **Fel 1: Vi saknar #include <stdio.h>.**
  Kompilatorn: Call to undefined function 'printf'
- **Fel 2: Vi har glömt & framför x vid inläsningen:**
  Kompilatorn: Possible use of x before definition
- **Fel 3: Det ska stå %f när vi läser in flyttal**
  Kompilatorn klagar inte. När programmet körs läses inget in, men programmet går att köra.
- **Fel 4: Det saknas %f i formatsträngen när vi ska skriva ut sum.**
  Värdet på sum skrivs inte ut.
- **Fel 5: Vanligt nybörjarfel: if-satsen är en tom sats.**
  Båda satserna körs alltid, den tomma if-satsen, som inte har någon effekt, och printf-satsen, som inte hänger ihop med if-satsen alls, if-satsen är avslutad i och med semikolonet. Rätt kod är:
  if(x<0)printf("X är negativt");

De första två felen var syntaktiska. De andra tre var logiska fel. Logiska är mer lömska än syntaktiska eftersom kompilatorn inte tillåter oss att köra ett syntaktiskt felaktigt program.