Loopar och datatyper

Föreläsning 3

Dagens kluringar

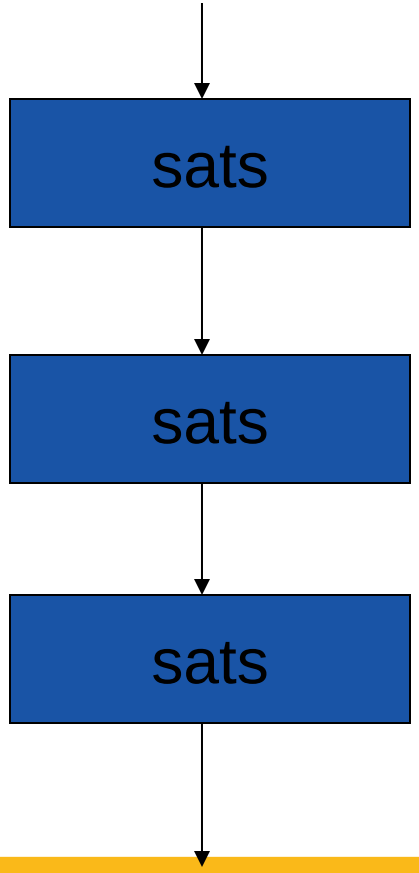
```
int x;
printf("Ange x:");
scanf("%d",&x);
if(/*fyll i kod*/)
    printf("Du angav x mellan 7 och 14");

int i=0;
if(i++)
    i++;
printf("%d",i++); //vad skrivs ut?
```

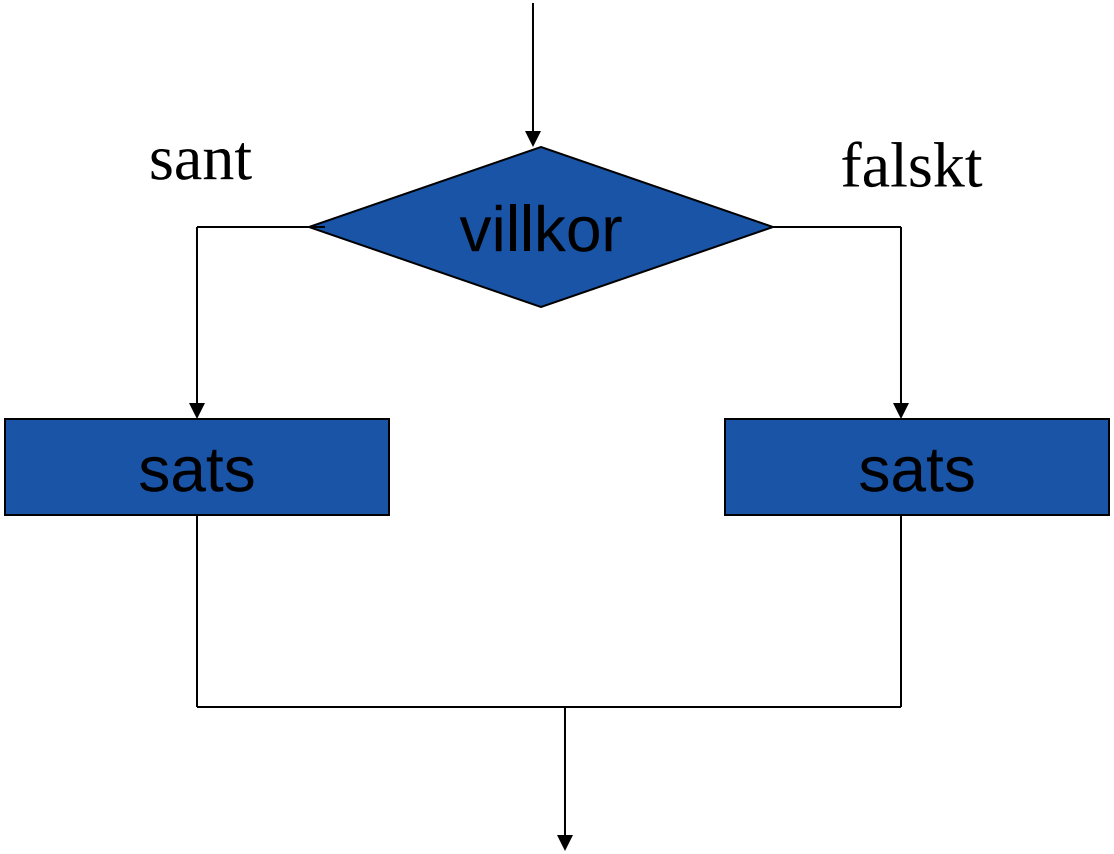
Loopar och datatyper

- Flödesscheman
- while
- for
- Datatyper
- Konvertering

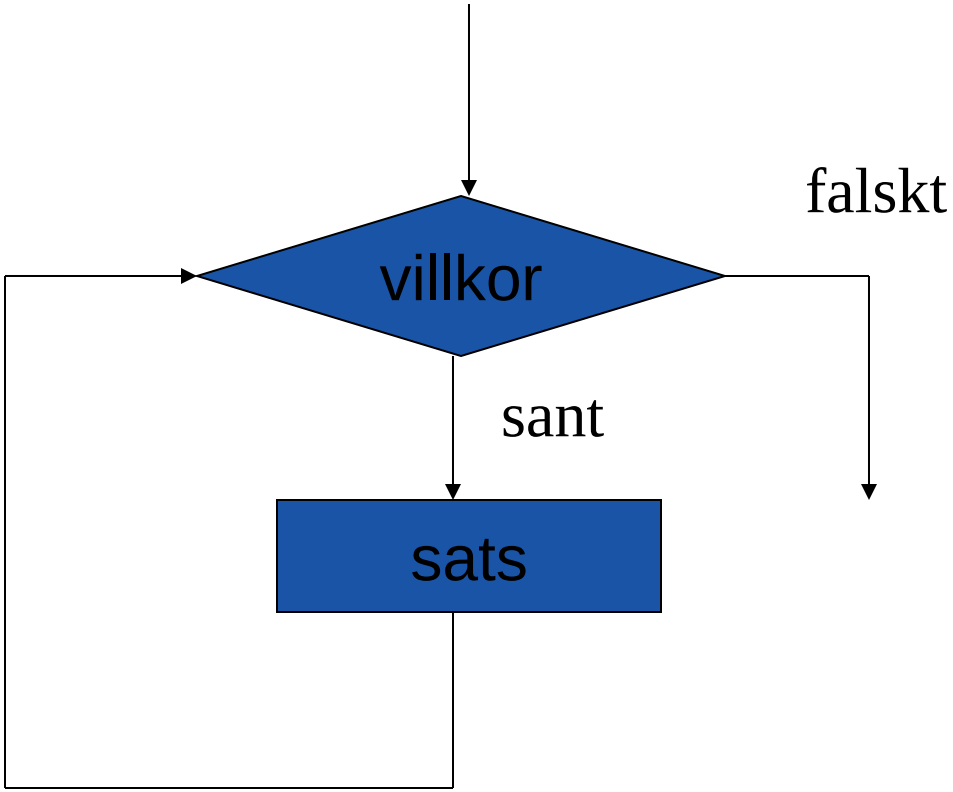
Sekvens



Selektion (if)



Iteration (for, while)



while

while betyder medans eller så länge

while(*villkor*)
{
sats1
sats2
…
}

-satserna upprepas så länge villkor är sant

Några exempel

```
while(1)
{
      printf(" det roliga tar aldrig slut\n ");
}
//ctrl-c i terminal, build->abort i codeblocks

int i = 0;
while(i<10)
{
      printf("%d, ", i);
      i=i+1;
}
```

OBS! ej: while(i<10);

```c
#include <stdio.h>

int main(void)
{
    int n=1;
    while(n != 0)
    {
        scanf("%d",&n);
        printf("Du skrev:%d \n", n);
    }
    return 0;
}
```

Summa-exempel

```
//Summera heltal

#include <stdio.h>

int main(void)
{
    printf("Summera heltal\n");
    printf("Skriv in heltal, avsluta med 0\n");

    int n=1, summa=0;

    while(n != 0)
    {
        scanf("%d",&n);
        summa = summa + n;
    }

    printf("Summa:%d \n", summa);
    return 0;
}
```

do - while

```
do
{
        sats1
        sats2
        …
} while(villkor);
```

- jämförelsen görs sist
- körs minst en gång
- ibland naturligare, tex i exemplen nyss

```c
#include <stdio.h>

int main(void)
{
    int n; //vi slipper initiera

    do
    {
        scanf("%d",&n);
        printf("Du skrev:%d \n", n);
    }while(n != 0);

    return 0;
}
```

for

for – emedan, så länge

används främst när vi vet hur många gånger vi vill upprepa något

Exempel på for-slinga (loop)

Startsats, utförs först och endast en gång

Utförs sedan och först i varje påföljande varv. Om resultatet blir true utförs {}.

```
int n;
for (n = 0;n<10;n++)
{
    printf("%d, ", n);
}
```

Utföres sist efter {} i varje varv. Används normalt för att stega upp eller ner n.

Resultatet blir: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,

Fler exempel

```
int n;
for(n=10;n>0;n--)
{
      printf("%d, ", n);
}

for(;1;)
{
     printf("evig loop\n");
}
```

```
int k;
int summa = 0;
for(k=1;k<=7;k=k+1)
{
      summa = summa + k;
      printf("Summa hittills: %d\n", summa);
}
```

En körning ser ut så här:

```
Summa hittills: 1
Summa hittills: 3
Summa hittills: 6
Summa hittills: 10
Summa hittills: 15
Summa hittills: 21
Summa hittills: 28
```

Nästlade loopar

```
int i,j;
for(i=1;i<=5;i=i+1)
{
      for(j=1;j<=3;j=j+1)
      {
            printf("(%d,%d) ", i, j);
      }
      printf("\n");
}
```

Resultat:
(1,1) (1,2) (1,3)
(2,1) (2,2) (2,3)
(3,1) (3,2) (3,3)
(4,1) (4,2) (4,3)
(5,1) (5,2) (5,3)

Max/Min av heltal

```
#include <stdio.h>
int main(void){
        int max,min,k,tal;
        printf("Tal nr 1: ");
        scanf("%d",&tal);
        min=tal;
        max=tal;
        for(k=2;k<=5;k++){
                printf("Tal nr %d: ",k);
                scanf("%d",&tal);
                if (tal<min) min=tal;
                if (tal>max) max=tal;
        }
        printf("%d %d\n",max,min);

        return 0;
}
```

Resultat:
Tal nr 1: **100**
Tal nr 2: **50**
Tal nr 3: **-40**
Tal nr 4: **2**
Tal nr 5: **40**
100 -40

Grundläggande datatyper

Vi har hitintills använt variabler av datatypen int (heltal) och float (decimaltal) för att lagra data. Även om vi främst ska använda dessa är det viktigt att veta att det finns fler datatyper om man tex behöver lagra större tal än som får plats i dessa. Vi ska också titta på datatypen char som används för att lagra bokstäver.

Heltal

Hur många bitar och därmed hur stora tal som kan representeras med datatyper i C är implementationsberoende (varierar). Vanliga värden på en 32-bitars maskin för de olika heltalstyperna:

Typ	minsta värde	största värde	printf,scanf
short int	-32 768	32 767	%hd
unsigned short int	0	65 535	%hu
int	-2 147 483 648	2 147 483 647	%d
unsigned int	0	4 294 967 295	%u
long int	-2 147 483 648	2 147 483 647	%ld
unsigned long int	0	4 294 967 295	%lu

Med en header som heter <limits.h> kan man ta reda på dessa värden i ett program. Pröva gärna vad som händer om man försöker stoppa in ett för stort tal.

Decimaltal

Decimaltal har både fördelen att de kan ha en decimaldel och att de har plats för större tal tack vare exponenten. Detta på bekostnad av att de inte är exakta. Det finns tre typer: float, double och long double.

Typ	Minsta positiva värde	Största värde	Presicion	printf, scanf
float	1.17549e-38	3.40282e38	6 siffror	%f, %f
double	2.22507e-308	1.79769e308	15 siffror	%f, %lf

Observera att $3{,}7 \cdot 10^{-2}$ skrivs 3.7e-2

Bokstäver

I datatypen **`char`** lagrar vi bokstäver. Egentligen lagras de som tal som tolkas som bokstäver enligt ASCII (sid 801 i King).

```
char b1, b2;
b1 = 'a';
b2 = 97;
printf("b1: %c, %d; b2: %c, %d\n",b1,b1,b2,b2);
```

Resultat:
b1: a, 97; b2: a, 97

I utökad ASCII finns fler tecken såsom åääö. Vi kommer oftast nöja oss med de tecken som finns i ASCII.

Exempel

Oftast struntar vi att char egentligen innehåller ett tal och tänker på char som att den lagrar en bokstav men ibland kan det vara användbart:

```
char ch;
scanf("%c",&ch);
if('a' <= ch && ch <= 'z')
{
    ch = ch - 'a' + 'A';
}
printf("%c",ch);
```

Resultat:

g
G

Ett subtilt problem

När man ska läsa in bokstäver uppstår ett subtilt problem…

```
char ch='a';
while(ch!='P')
{
    printf("Ange bokstav:");
    scanf("%c",&ch);
    if('a' <= ch && ch <= 'z')
    {
        ch = ch - 'a' + 'A';
    }
    printf("%c\n",ch);
}
```

```
Ange bokstav:b
B
Ange bokstav:

Ange bokstav:c
C
Ange bokstav:

Ange bokstav:p
P
```

Förklaring

Problemet beror på att även <enter> är ett tecken, nämligen ’\n’. När du slår in <a> <enter> läggs ’a’ och ’\n’ i en ström redo att läsas. scanf läser första tecknet ’a’ men ’\n’ ligger kvar och läses nästa varv när scanf exekveras. Detta har inte varit ett problem tidigare då scanf hoppar över whitespaces (enter, mellanslag, tab,…) när den läser in heltal eller decimaltal. Vi kommer senare att lära oss mer om inläsning. Här kommer ett förslag på primitiv lösning så länge:

```
char ch='a',ch_tmp;
while(ch!='P')
{
    printf("Ange bokstav:");
    scanf("%c",&ch);
    scanf("%c",&ch_tmp);
    if('a' <= ch && ch <= 'z')
    {
        ch = ch - 'a' + 'A';
    }
    printf("%c\n",ch);
}
```

En annan lösning som klarar att användaren dräller med mellanslag men som avslutar med att lämna ett enter:

```
char ch='a',ch_tmp;
    while(ch!='P')
    {
        printf("Ange bokstav:");
        scanf(" %c",&ch); //mellanslag före %c
        if('a' <= ch && ch <= 'z')
        {
            ch = ch - 'a' + 'A';
        }
        printf("%c\n",ch);
    }
```

Konvertering

(`typ`) – denna operator konverterar om möjligt talet till höger till datatypen `typ`.

```
int i = (int) 3.45;
```

Konverteringen sker dock automatiskt vid behov:

```
int i = 3.45;//ger samma resultat

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

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

Studieanvisningar F3

- Skriv tre while-loopar som skriver ut följande:
  a) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
  b) 10,9,8,7,6,5,4,3,2,1,
  c) 1, 2, 4, 8, 16,
- Skriv tre for-loopar som skriver ut ovanstående.
- Läs 6.1-6.3 (testa på datorn samtidigt som du läser)
- Gör om cd-skivsprogrammet så att användaren efter att ha fått reda på priset får frågan om hon vill göra ett nytt köp eller avsluta. Om användaren vill göra ett nytt köp ska programmet köra igen. På så sätt kan man göra valfritt antal köp innan programmet avslutar.
- Gör K6 P1, P2, P3 (sid 122-123)
- Skumma igenom 7.1 och 7.2. Läs om oktala och hexadecimala tal sid 128 ordentligt om du inte kan detta.
- Läs 7.3 sid 134-135 och 7.3 andra halvan av sid 139
- Gör K7 P15 (fakultet: $4! = 1 \cdot 2 \cdot 3 \cdot 4$)
- Gör fler uppgifter om du hinner
- Svara på instuderingsuppgifter

E-excercises, P-programming projects