Switch, Array (fält)

- switch
- break, continue, *goto* (scary)
- Sammansatta tilldelningar
- Kommentarer
- Array
- Sortering

switch

```
int weekday;
printf("Mata in veckodagnummer 1-7: ");
scanf("%d", &weekday);
switch(weekday)
{
	case 1: printf("Monday\n");
	break;
	case 2: printf("Tuesday\n");
	break;
	case 3: printf("Wednesday\n");
	break;
	case 4: printf("Thursday\n");
	break;
	case 5: printf("Friday\n");
	break;
	default: printf("Weekend!\n");
}
```

jmf:

```
if(weekday==1)
		printf("Monday\n");
else if(weekday==2)
		printf(“Tuesday\n");
else if(weekday==3)
		printf(“Wednesday\n");
else if(weekday==4)
		printf(“Thursday\n");
else if(weekday==5)
		printf(“Friday\n");
else
		printf(“Weekend! \n ");
```

break

- Skrivs inuti en switch, for, while eller do-while sats
- När den exekveras hoppar programmet genast ut ur den sats (switch, for, while, do-while) den står i
- Med undantag för switch-satsen behöver den normalt inte användas

```
int i=1,summa=0,tmp;
   for(i=0;i<10;i++)
   {
      printf("Skriv in tal %d:",i);
      scanf("%d",&tmp);
      if(tmp==0)
         break;
      else
         summa = summa+tmp;
   }
   printf("Summa:%d",summa);
```

continue

- Skrivs inuti en for, while eller do-while sats
- När den exekveras påbörjas ett nytt varv
- Finns egentligen aldrig en bra anledning att använda denna

goto

- **Använd aldrig under några omständigheter denna sats!**
- Om ni någon gång använder goto se då till att följa regeln ovan
- Möjligen kan användandet av goto tillfälligt under debugging vara motiverat men jag tycker att även här använder vi regel ett för att reglera användandet

Den sammanlagda kostnaden för världen pga goto hann innan användandet begränsades överstiga kostnaden för NASA och CERN tillsammans!

(källa: ingen alls men jag känner på mig att det stämmer)

Sammansatta tilldelningar

a+=4	motsvarar	a=a+4
a-=4	motsvarar	a=a-4
a*=4	motsvarar	a=a*4
a/=4	motsvarar	a=a/4

-fungerar även med andra tal än 4 ☺

Kommentarer

- Finns av två typer:
  //resten av raden ignoreras av kompilatorn
  /*kan löpa över flera rader och avslutas
  vid*/
  OBS att den senare inte kan nästlas!
- Skall förklara koden men inte förklara sådant som man förstår om man förstår C. Man ska ha lagom många kommentarer ! Endast genom övning lär man sig vilka kommentarer som behövs och vilka som är överflödiga.
- Nästa lab försök skriva de kommentarer du tror du behöver för att enkelt förstå koden om två veckor. Läs koden två veckor senare och se om det hade behövts fler kommentarer och om några var överflödiga.

Preprocessor (förprocessor)

```
#include <stdio.h>

#define MOMS 25
#define RABATT 10

int main(void) {
    int antal=10;
    float tpris;
    const float pris=3.25;
    tpris=antal*pris*(1+MOMS/100.0)*(1-RABATT/100.0);
    printf("%6.2f kr\n",tpris);
    getch();
}
```

Indicerade variabler-
Fält -Array

- Används för att lagra flera instanser av samma sorts data, tex tärningskast, personnummer, namn
- Med en for-loop kan man sedan enkelt utföra samma operation på alla variabler i en array, tex summera, beräkna ålder, jämföra
- *Är det sista verktyget vi behöver bemästra innan vi kan skriva riktiga program!*

Ett menlöst exempel

```
int a[3];
a[0]=2;
 a[1]=7;
a[2]=9;
printf("Test: %d, %d, %d",a[0],a[1],a[2]);
```

OBS! – första platsen har index 0 och sista har index 2 om antalet platser är 3

Array med for

```
int langd=8;
int a[langd];
int i;
for(i=0;i<langd;i++)
{
   a[i]=i+1;
}
for(i=0;i<langd;i++)
{
   printf("Pa plats: %d har vi %d\n",i,a[i]);
}
```

```
Pa plats: 0 har vi 1
Pa plats: 1 har vi 2
Pa plats: 2 har vi 3
Pa plats: 3 har vi 4
Pa plats: 4 har vi 5
Pa plats: 5 har vi 6
Pa plats: 6 har vi 7
Pa plats: 7 har vi 8
```

Ett till exempel

```
#include <stdio.h>
int main(void)
{
    int s[13],kast,k;
    srand(time(0));
    for(k=2;k<=12;k++)
        s[k]=0;
    for(k=1;k<=10000;k++)
    {
        kast=rand()%6+rand()%6+2;
        s[kast]++;
    }
    for(k=2;k<=12;k++)
     printf("Antal %2d: %d. Andel: %.2f%%\n",k,s[k],100*s[k]/10000.);
}
```

```
Antal  2: 318. Andel: 3.18%
Antal  3: 505. Andel: 5.05%
Antal  4: 854. Andel: 8.54%
Antal  5: 1129. Andel: 11.29%
Antal  6: 1395. Andel: 13.95%
Antal  7: 1704. Andel: 17.04%
Antal  8: 1365. Andel: 13.65%
Antal  9: 1112. Andel: 11.12%
Antal 10: 823. Andel: 8.23%
Antal 11: 516. Andel: 5.16%
Antal 12: 279. Andel: 2.79%
```

Sortering

Även om datorer idag används till väldigt mycket olika saker är fortfarande två av de viktigaste funktionerna att lagra data och att tillhandahålla valda delar av dessa data.

Mycket forskning har gjorts i hur detta skall göras på bästa/effektivast sätt.

Målet är att man så fort som möjligt ska kunna få tag på valda delar av lagrade data.

För att hitta rätt data behöver man **söka** igenom tillgänglig data.

Detta kan effektiviseras med bra sökalgoritmer och genom **sortering** av data i förväg.

(Allra effektivast blir det om man använder nycklar och tabeller för att organisera data – relations-databaser)

Bubbelsortering

Vi ska börja med att titta på hur man kan sortera data. I exemplet tittar vi på hur man sorterar heltal men metoden fungerar lika bra på alla sorters data som kan ordnas i en bestämd sekvens, t.ex decimaltal, bokstäver, ord.

Algoritmen vi ska titta på kallas bubbelsortering eftersom de större talen tillåts bubbla upp (till höger).

Steg 1 (*14 bubblar upp*)

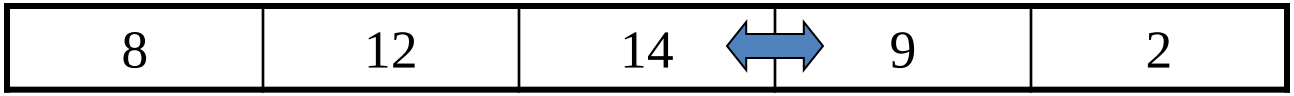
Jämför talen två och två från vänster till höger. Om talet till vänster är större byter man plats.1. Jämför tal 0 och 1:

Talet till vänster är större så vi byter plats!

8	12	14	9	2

Nu jämför vi tal 1 och 2:

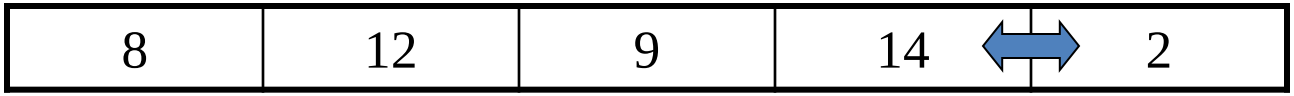
Talet till vänster är mindre så ingen åtgärd. Nu jämför vi tal 2 och 3:



Talet till vänster är större så vi byter plats:

8	12	9	14	2

Nu jämför vi tal 3 och 4:



Talet till vänster är större så vi byter plats:

8	12	9	2	14

Resultat: Steg 1

Vad har vi då åstadkommit?

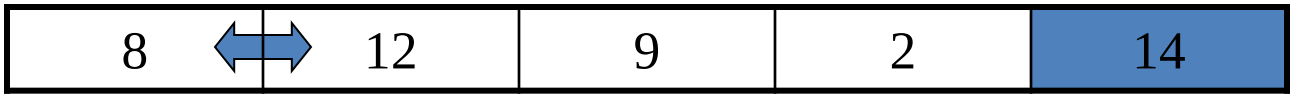
Jo vi kan vara säkra på att det största talet befinner sig längst till höger. Detta gäller oberoende hur det såg ut från början.

Övertyga gärna dig själv om detta!

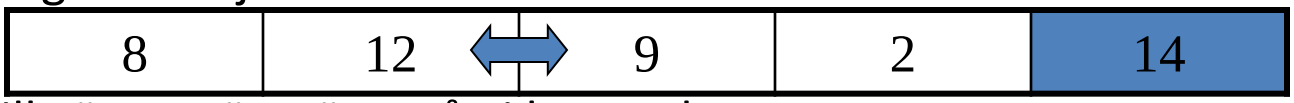
Vad gör vi nu? Jo samma sak med fältet utom den sista platsen.

Steg 2 (*12 bubblar upp*)

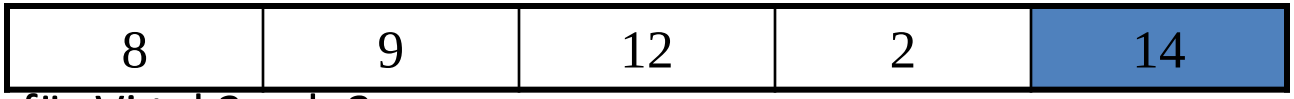
Jämför tal 0 och 1:



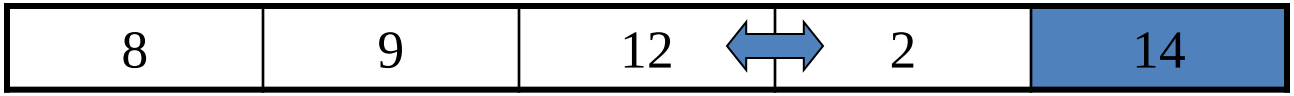
Ingen åtgärd. Nu jämför vi tal 1 och 2:



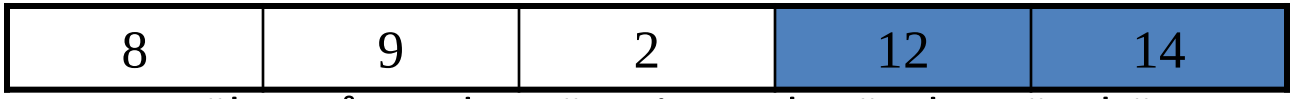
Talet till vänster är större så vi byter plats:



Nu jämför Vi tal 2 och 3:



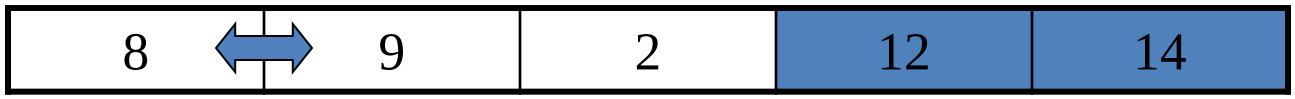
Talet till vänster är större så vi byter plats:



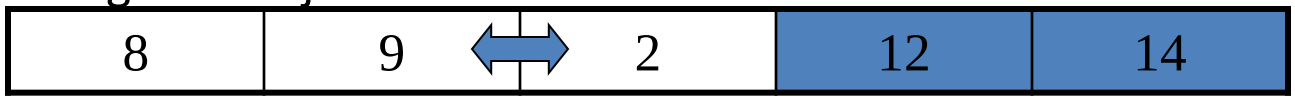
Vi kan nu vara säkra på att det näst sista talet är det näst högsta. Nu behöver vi bara upprepa algoritmen två gånger till för att vara säkra på att alla tal kommer i nummerordning.

Steg 3 och 4

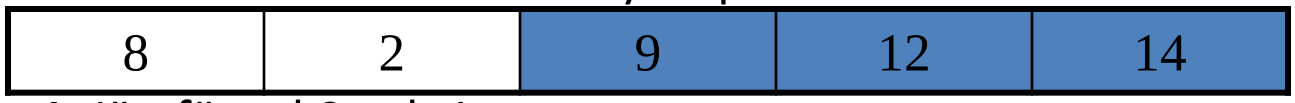
Steg 3: Jämför tal 0 och 1:



Ingen åtgärd. Nu jämför vi tal 1 och 2:



Talet till vänster är större så vi byter plats:



Steg 4: Jämför tal 0 och 1:

Talet till vänster är större så vi byter plats:

Vi kan nu vara säkra på att alla tal är i nummerordning!

Algoritm i pseudokod:

Sortera fält **f** med antal element **n**:

För **i** från **0** till **n – 2**:

För varje varv bubblar ett nytt tal upp.

För **j** från **0** till **n – 2 – i:**

Den inre loopen bubblar upp ett tal.

Om **f[j]>f[j+1]** byt plats

```
for(i=0;i<antal-1;i++)
{
      for(j=0;j<antal-1-i;j++)
      {
         if(tal[j]>tal[j+1])
         {
               tmp=tal[j];
               tal[j]=tal[j+1];
               tal[j+1]=tmp;
         }
      }
}
```

Varför fungerar inte:
tal[j]=tal[j+1]
tal[j+1]=tal[j]

```
int main()
{
   int antal=10;
   int tal[antal];
   srand(1);
   int i;
   for(i=0;i<antal;i++)
   {
      tal[i]=rand()%20;
   }
   skrivUt(tal,antal);
   printf("\n");
   int j,tmp;
   for(i=0;i<antal-1;i++)
   {
      for(j=0;j<antal-1-i;j++)
      {
         if(tal[j]>tal[j+1])
         {
               tmp=tal[j];
               tal[j]=tal[j+1];
               tal[j+1]=tmp;
         }
      }
   }
   skrivUt(tal,antal);
}
```

```
void skrivUt(int x[],int langd)
{
   int i;
   for(i=0;i<langd;i++)
   {
      printf("%d,",x[i]);
   }
}
```

Matriser

```
int rad=3,kolumn=4;
int a[rad][kolumn];
int i,j;
int plats=1;
for(i=0;i<rad;i++)
{
   for(j=0;j<kolumn;j++)
   {
      a[i][j]=plats;
      plats++;
   }
}
```

```
 1, 2, 3, 4
 5, 6, 7, 8
9,10,11,12
```

Sista ord

En av de svåraste delarna av kursen

Jobba hårt med att förstå array och sortering

Gör många uppgifter på kap 6 i boken