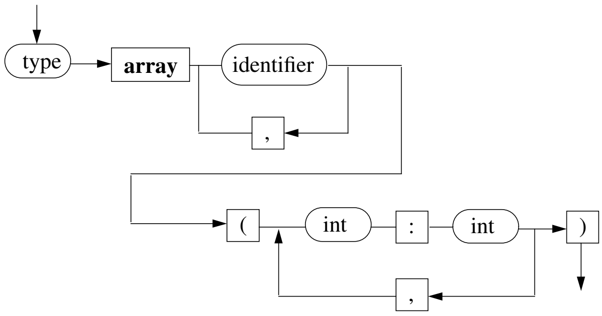
Tabeller

type **array** navn (x:y)



character array n (1:10)
text array s (1:10)
integer array m (1:10, 2:5)
real array w (1:10, 2:5, 7:19)
integer array n1, n2, n3 (1:10, 2:5)

Programmønstre – noen enkle algoritmer

- **Finn minste verdien i en tabell**

```
integer array tabell(lav:hoy);
integer minste, teller;

minste := tabell(lav);
for teller:=lav+1 step 1 until hoy do
   minste := min(minste, tabell(teller));
```

- **Finn minste verdien i en 2-dimensjonal tabell**

```
integer array tabell(lav1:hoy1, lav2:hoy2);
integer minste, teller1, teller2;

minste := tabell(lav1, lav2);
for teller1:=lav1 step 1 until hoy1 do
   ! Finn minste i tabell(teller1,...) ;
   for teller2:=lav2 step 1 until hoy2 do
      minste := min(minste, tabell(teller1, teller2));
```

- **Finn en gitt *verdi* (om den finnes!) i en tabell**

```
integer array tabell(lav:hoy);
integer verdi, teller; boolean funnet;

teller := lav; funnet := false;                    ! begynn øverst i tabellen ;
while teller<=hoy and not funnet do
   if tabell(teller)=verdi then funnet:=true             ! verdi er funnet ;
                          else teller:=teller+1;    ! ellers fortsett nedover ;

! Nå gjelder det at: if funnet then tabell(teller)=verdi else teller=hoy+1;
```

Enkle algoritmer (forts.)

- **Finn en gitt verdi *v* (om den finnes!) i en sortert tabell**

(Binary Search)

```
integer array A(lav:hoy);
integer l, h, m; boolean funnet, finnes

l := lav; h := hoy;
funnet := false; finnes := true;
while finnes and not funnet do
   begin
      m := (l + h) // 2;                  ! sjekk i m - midten av tabellen ;
      if A(m) = v then funnet := true     ! v er i A(m) ;
      else if l > h then finnes := false  ! v finnes ikke ;
      else if A(m) < v then l := m+1      ! finnes v, sa er den i ovre halvparten ;
      else                  h := m-1 ;    ! finnes v, sa er den i nedre halvparten ;
   end ;

! Nå gjelder det at: if funnet then A(m) =v else not finnes;
```


Insertion Sort

1	2	3	4	5
8	4	6	2	1

<sorter elementene 1-2>

1	2	3	4	5
4	8	6	2	1

<sett inn el. 3 paa riktig plass>
<s.a. elementene 1-3 er sortert>

1	2	3	4	5
4	6	8	2	1

< sett inn el. 4 paa riktig plass>
< o. s. v. >

1	2	3	4	5
2	4	6	8	1

< inntil du har gått gjennom hele tabellen>

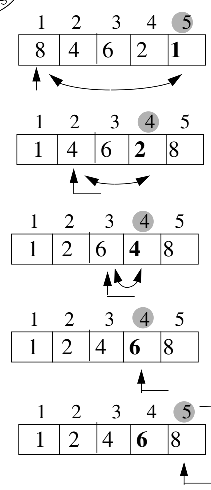
for** p:= lav+1 **step** 1 **until** hoy **do
   sett *A(p)* inn i riktig posisjon *j*
   bland elementene *A(lav)...A(p)*

f.eks. < finn riktige posisjonen *j* for *A(p)*:
   lav <= j <= p >
   < flytt *A(j)... A(p-1)* en pos. til hoyre >
   < plasser *A(p)* i posisjonen *j* >

```
integer array A(lav:hoy) ;
integer tmp ;
integer p, j ;

for p:=lav+1 step 1 until hoy do
  begin
    j:= p; tmp := A(p);
    while j > lav and then
                  tmp < A(j-1) do
       begin
          A(j) := A(j-1);
          j:= j-1;
       end;
    A(j) := tmp;
  end
```

Selection Sort



<finn minste i hele tabellen>
<bytt tabell(1) med den minste>

<fortsett fra og med tabell(2)>
<finn minste i tabell(2) til tabell(5)>
<bytt tabell(2) med den minste>

< o. s. v. >

< inntil du har gått gjennom hele tabellen>

minste_ind

ind

```
integer array tabell(lav:hoy);
integer ind, minste_ind, tmp, k;

for ind:=lav step 1 until hoy-1 do
begin
    minste_ind := ind;
    for k:=ind+1 step 1 until hoy do
      if tabell(k) < tabell(minste_ind) then minste_ind:=k ;

    tmp := tabell(minste_ind);
    tabell(minste_ind) := tabell(ind);
    tabell(ind) := tmp;
end
```