Imperativ programmering

1DL126 3p

Föreläsning 2

Imperativ programmering

Kännetecken för imperativa språk:

- Programmet består av en serie instruktioner.

Olika språk har olika uppsättningar av instruktioner. Dessa kallas helt oväntat för instruktions-uppsättningar.

Instruktioner

- Tre typer:
  - Tilldelning
  - Operatorer
  - Kontrollflöde

Imperativ programmering

Kännetecken för imperativa språk:

- Programmet består av en serie instruktioner.
- Instruktionerna förändrar omgivningens tillstånd.

Detta sker genom tilldelning.

Vad är en tilldelning?

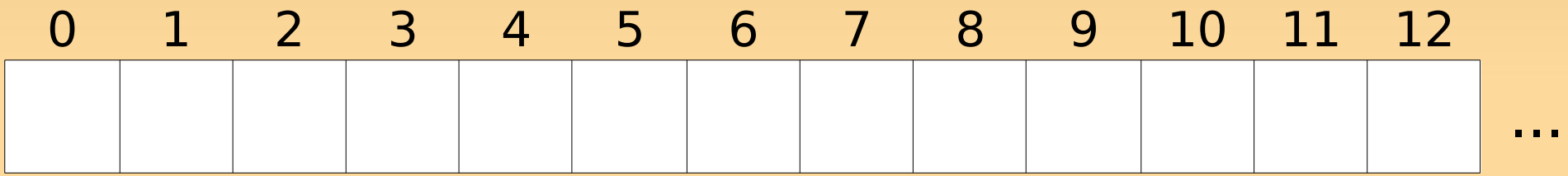
- En tilldelning förändrar ett värde i maskinen - maskinens tillstånd förändras.

- Tillstånd - Ett foto av maskinens minne. Tillståndet ärver sin betydelse direkt från värddatorn i ett imperativt språk.

Minnet

- Minnet består av en serie platser.
- En adress är numret på en plats i minnet.
- Varje plats kan innehålla ett tal.
- Talet som ligger på en plats i minnet kallas *innehållet* på den adressen.

Minnet



Minnet

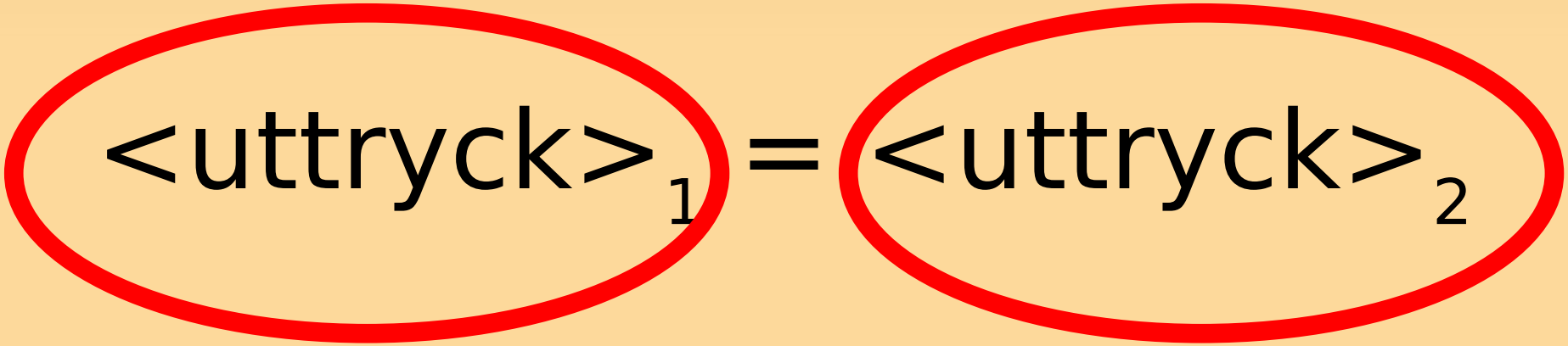
0	1	2	3	4	5	6	7	8	9	10	11	12	
42	127	0	76	2	208								...

0	1	2	3	4	5	6	7	8	9	10	11	12	
75	3	35	87	63	12	52	4	98					...

Tilldelning

L-value

R-value



Plats i minnet

Värde som kan lagras där

L-value och R-value

M[j]

L-value - En plats (j)

R-value - Innehållet

Adressering

M[j] - Direkt adressering

M[M[j]] - Indirekt adressering

R-värdet från M[j] är L-värde för M[M[j]]

Bindning

- Att binda - Associera ett namn med en plats i minnet.
- Namnet refererar till platsen.
- En abstraktion i programmeringsspråken.

Bindning

- Namnet binds vid första användning:

  x = 42;

- Namnet måste bindas innan användning:

  (Deklaration)

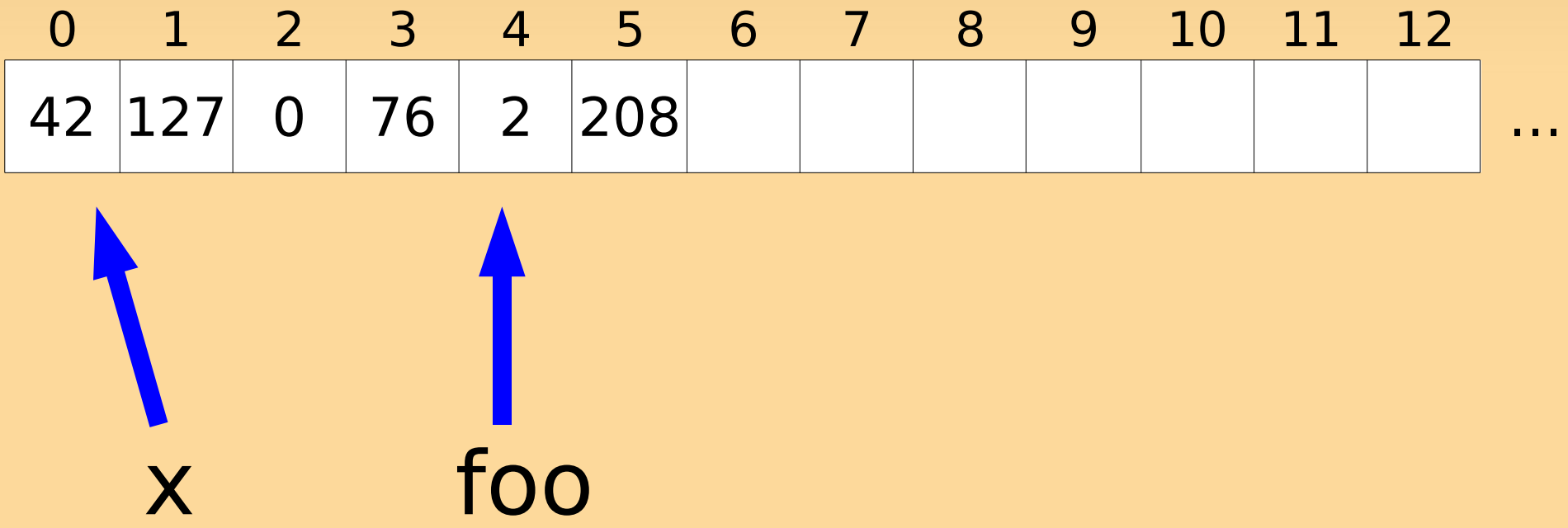
  var x;

  x = 42;

- Deklaration kan även specificera typ:

  int x; var x : integer;

Bindning



Bindning

- Statisk bindning - sker innan programmet körs.
- Dynamisk bindning - sker medan programmet körs.

Statisk bindning

- Funktioner i C - Koden kan inte ändras under körning.

```
void foo() { blah = 1; }

void bar() { foo(); }
```

Dynamisk bindning

- Generisk programmering i t.ex. Java, Ada, C++.

```
public void foo(List<String> l) {
    l.add("bar");
}
```

Bindning

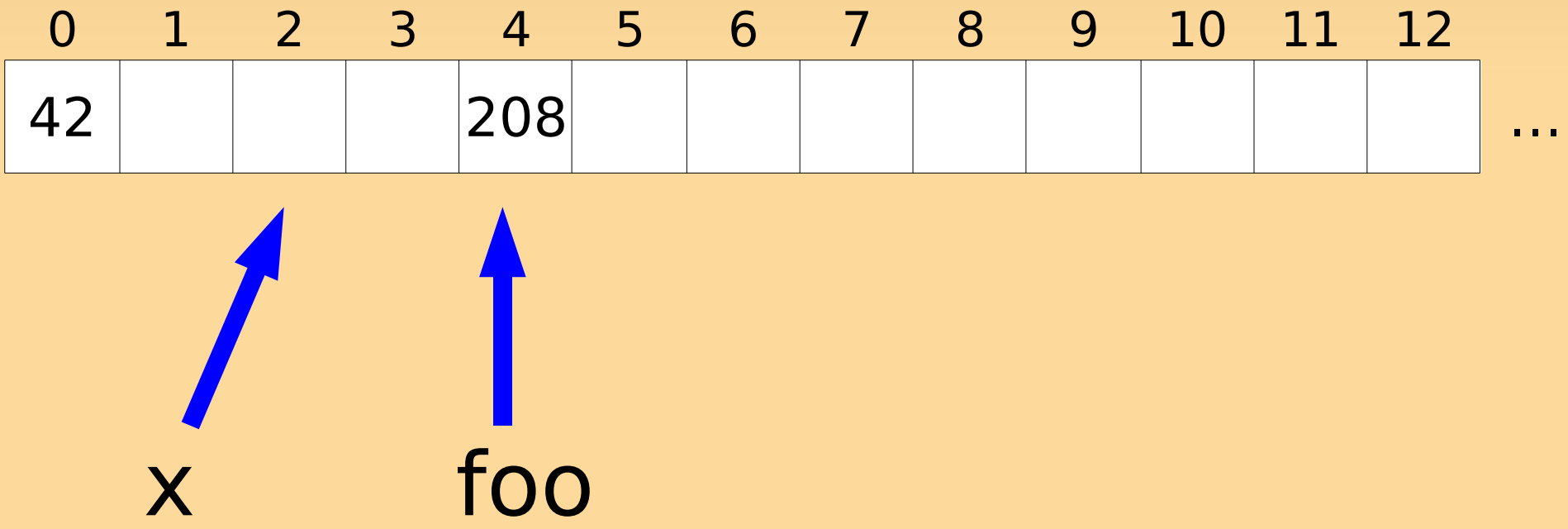
- Vi kan binda om ett namn så att det refererar till en annan plats i minnet.
- OBS! Inget ändras i själva minnet.

Bindning

```
var x = 42, foo = 208;
```

Bindning

```
x = &M[2];
```



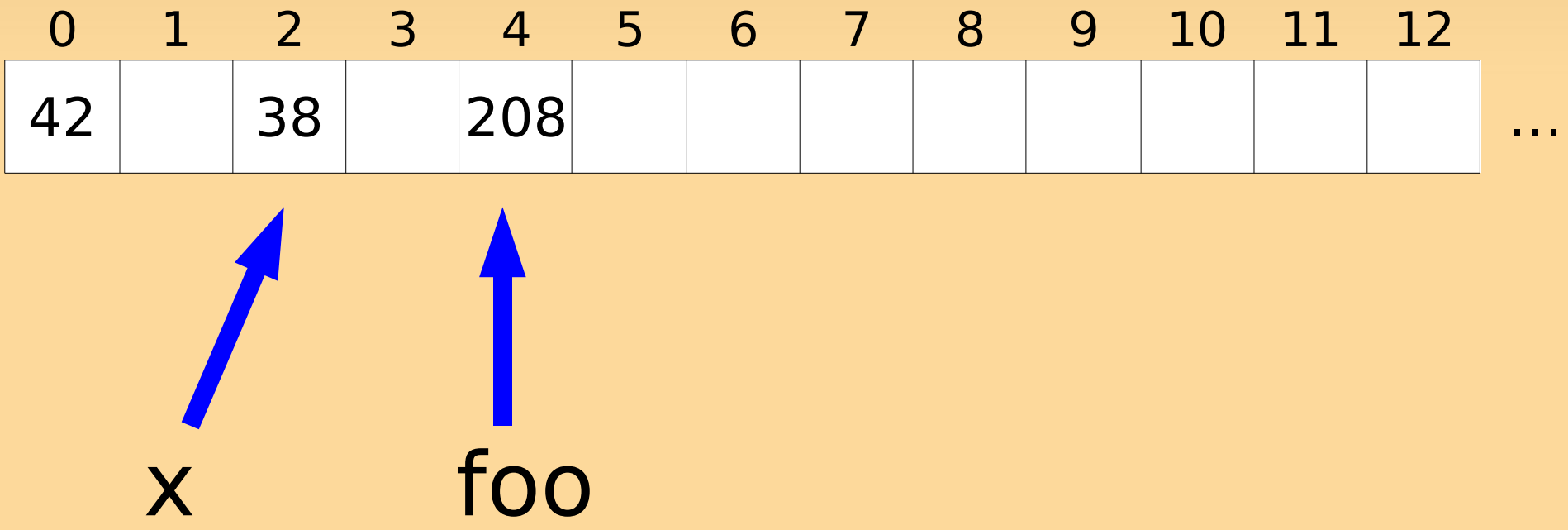
Bindning

En viktig skillnad mellan imperativa språk och funktionella språk: Imperativa språk kan ändra i bundet minne.

Kallas att mutera.

Bindning

```
*x = 38;
```



Instruktioner

- Tre typer:
  - Tilldelning
  - Operatorer
  - Kontrollflöde

Operatorer

- Grundläggande funktioner i språket.
  - Matematiska operationer
  - Booleska operationer
  - Strängoperationer
  - Minnesoperationer

Operatorer

- Tre typer:
  - Unära: !x, &x, *x, ~x, @x, ^x
  - Binära: x + y, x - y, x * y, x / y
  - Namngivna: sizeof(x)

Operatorer

- Överlagring - Samma operator kan användas till flera olika saker.
- tal + tal sträng + sträng
- OBS! Inte tal * tal *var
- C++ tillåter egen överlagring av operatorer.

Instruktioner

- Tre typer:
  - Tilldelning
  - Operatorer
  - Kontrollflöde

Kontrollflöde

- Instruktioner som används för att styra programmet. Påverkar inte tillståndet i maskinen.
  - Villkorliga hopp
  - Ovillkorliga hopp
  - Loopar

Kontrollflöde

Villkorliga hopp:

C / C++ / Java

```
if (x == 10)
    sats;

if (x > 10)
    sats;
else if (x < 10)
    sats;
```

Ada

```
if x = 10 then
   sats;

if x > 10 then
   sats;
elsif x < 10 then
   sats;
```

Kontrollflöde

Villkorliga hopp:

C / C++ / Java

```
if (x == 10)
    sats;

if (x > 10)
    sats;
else if (x < 10)
    sats;
```

Pascal

```
if x = 10 then
   sats;

if x > 10 then
   sats
else if x < 10 then
   sats;
```

Kontrollflöde

Switch / case

C / C++ / Java

```
switch (heltal) {
   case 54: satser; break;
   case 78: satser; break;
   default: satser;
}
```

Kontrollflöde

Ada:

```
case Heading is
   when North => Y := Y + 1;
   when South => Y := Y  - 1;
   when East => X := X + 1;
   when West => X := X - 1;
end case;
```

Kontrollflöde

Ovillkorliga hopp:

goto, longjmp, gosub

- Procedurell programmering förbjuder ovillkorliga hopp (bortsett från return, break, exit, continue...)
- Java har inte goto

Kontrollflöde

Loopar - Fasta

```
for (x = 0; x < 10; x++)

for x in integer range 0 .. 10 loop
end loop                          (reverse)

for i := 0 to 10 do               (downto)
```

Kontrollflöde

For-each (Java):

```
for (Object x : list)
   foo(x);
```

For över ett intervall (Ada):

```
for elm in arr'range loop
   arr(elm) := 0;
```

Kontrollflöde

Loopar - dynamiska

```
while (antal < 10)


do {
} while (antal < 10);


repeat
until antal = 10;
```

Kontrollflöde

Loopar i Ada

```
loop
end loop;


loop
   exit when x = 10;
end loop;
```

Kontrollflöde

- Nyckelord i loopar
  - break
    - Avbryter loopen
  - continue
    - Hoppar tillbaka till början
  - exit (Ada)

Obligatorisk uppgift

Skriv ner det viktigaste / det du minns bäst från dagens föreläsning