

MiniMao (Modello Astratto Operazionale)

Costrutti iterativi alternativi

Altri costrutti iterativi

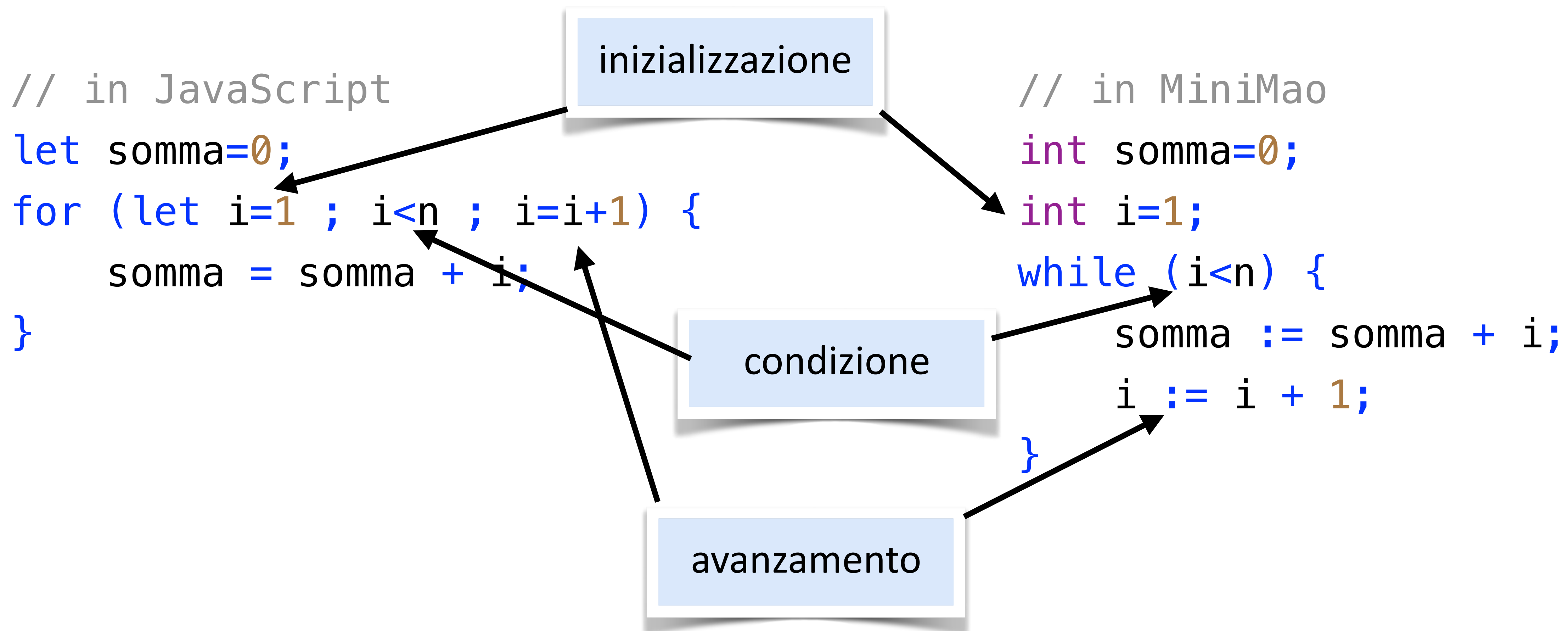
In Mao vediamo solo il costrutto `while (E) {C}` ma ci sono diversi costrutti iterativi che i linguaggi di programmazione mettono a disposizione

Ne vediamo brevemente alcuni perché saranno utili per scrivere programmi compatti in pseudocodice

Ognuno di questi costrutti **può essere espresso usando cicli while** e quindi non ne definiamo esplicitamente la semantica

Altri costrutti iterativi: il ciclo for

Il costrutto for è particolarmente compatto, di immediata lettura e informativo: in una riga sappiamo quante iterazioni sono necessarie (infatti ne raccomandiamo l'uso proprio in queste situazioni)



Altri costrutti iterativi: il ciclo do-while

Il costrutto **do-while** si usa quando il corpo del ciclo deve essere eseguito almeno una volta: prima eseguiamo il corpo e solo dopo valutiamo la guardia

```
// in JavaScript, col do-while
```

```
let età;
```

```
do {
```

```
    età = parseInt(prompt("Anni?"));
```

```
} while (età < 0)
```

```
// in JavaScript, col while
```

```
let età = parseInt(prompt("Anni?"));
```

```
while (età < 0) {
```

```
    età = parseInt(prompt("Anni?"));
```

```
}
```

Esercizi MiniMao

Esercizio: MCD Euclide differenze

Scrivi un programma che calcoli il Massimo Comun Divisore (MCD) di due numeri interi positivi utilizzando **l'algoritmo di Euclide delle sottrazioni successive**.

Descrizione dell'algoritmo, si prendono due numeri a e b ,

finché sono diversi tra loro:

- se $a > b$, *sostituisci a con $a-b$* ;
- altrimenti ($b > a$) *sostituisci b con $b-a$* .

Quando a e b diventano uguali, quel valore è il MCD.

MCD tra 72 e 32

a	b
72	32
$72-32= 40$	32
$40-32= 8$	32
8	$32-8= 24$
8	$24-8= 16$
8	$16-8= 8$

Esercizio: MCD Euclide differenze

Presi due numeri a e b , **finché sono diversi** tra loro:

- se $a > b$, *sostituisci a con $a-b$* ;
- altrimenti ($b > a$) *sostituisci b con $b-a$* .

Quando a e b diventano uguali, quel valore è il MCD.

```
while (.....) {
```

```
}
```

Esercizio: MCD Euclide con modulo

Scrivi un programma che calcoli il Massimo Comun Divisore (MCD) di due numeri interi positivi utilizzando **l'algoritmo di Euclide con il resto (accelerato)**.

Descrizione dell'algoritmo, si prendono due numeri a e b ,

finché b non è uguale a zero:

- calcola il resto $r = a \% b$,
- *sostituisci a con b e poi sostituisci b con r .*

Quando $b = 0$, il valore di a è il MCD.

MCD tra 72 e 32

a	b
72	32
32	$72 \% 32 = 8$
8	$32 \% 8 = 0$

Esercizio: MCD Euclide con modulo

Presi due numeri a e b ,

Finché b non è uguale a zero:

- calcola il resto $r = a \% b$,
- *sostituisci a con b e poi sostituisci b con r .*

Quando $b = 0$, il valore di a è il MCD.

```
while ( . . . . . ) {
```

```
}
```

Semantica operativa dei comandi MiniMao

$$\begin{array}{c}
 \frac{}{\langle \text{skip};, \rho, \sigma \rangle \rightarrow \langle \rho, \sigma \rangle} \quad \frac{\langle C_1, \rho, \sigma \rangle \rightarrow \langle \rho_1, \sigma_1 \rangle \quad \langle C_2, \rho_1, \sigma_1 \rangle \rightarrow \langle \rho_2, \sigma_2 \rangle}{\langle C_1 C_2, \rho, \sigma \rangle \rightarrow \langle \rho_2, \sigma_2 \rangle} \\
 \\
 \frac{\langle E, \rho, \sigma \rangle \Downarrow v \quad l \notin \sigma}{\langle \text{T Id} = E; , \rho, \sigma \rangle \rightarrow \langle \rho[\text{Id} \mapsto l], \sigma[l \mapsto v] \rangle} \quad \frac{\langle E, \rho, \sigma \rangle \Downarrow v \quad \rho(\text{Id}) = l}{\langle \text{Id} := E; , \rho, \sigma \rangle \rightarrow \langle \rho, \sigma[l \mapsto v] \rangle} \\
 \\
 \frac{\langle C, \rho, \sigma \rangle \rightarrow \langle \rho_1, \sigma_1 \rangle}{\langle \{C\}, \rho, \sigma \rangle \rightarrow \langle \rho, \sigma_1 \rangle} \quad \frac{\langle E, \rho, \sigma \rangle \Downarrow \text{true} \quad \langle C_1, \rho, \sigma \rangle \rightarrow \langle \rho_1, \sigma_1 \rangle}{\langle \text{if} (E) \{ C_1 \} \text{ else } \{ C_2 \}, \rho, \sigma \rangle \rightarrow \langle \rho, \sigma_1 \rangle} \\
 \\
 \frac{\langle E, \rho, \sigma \rangle \Downarrow \text{false}}{\langle \text{while} (E) \{ C \}, \rho, \sigma \rangle \rightarrow \langle \rho, \sigma \rangle} \quad \frac{\langle E, \rho, \sigma \rangle \Downarrow \text{false} \quad \langle C_2, \rho, \sigma \rangle \rightarrow \langle \rho_2, \sigma_2 \rangle}{\langle \text{if} (E) \{ C_1 \} \text{ else } \{ C_2 \}, \rho, \sigma \rangle \rightarrow \langle \rho, \sigma_2 \rangle} \\
 \\
 \frac{\langle E, \rho, \sigma \rangle \Downarrow \text{true} \quad \langle C, \rho, \sigma \rangle \rightarrow \langle \rho_1, \sigma_1 \rangle \quad \langle \text{while} (E) \{ C \}, \rho, \sigma_1 \rangle \rightarrow \langle \rho_2, \sigma_2 \rangle}{\langle \text{while} (E) \{ C \}, \rho, \sigma \rangle \rightarrow \langle \rho, \sigma_2 \rangle}
 \end{array}$$

Esercizio: cifre

Scrivi un programma che dato un intero positivo calcola la somma di tutte le sue cifre che sono multipli di 3

Esempio: 183657 ➡ 3 + 6 = 9

Esercizio: cifre

Scrivi un programma che dato un intero positivo calcola la somma di tutte le sue cifre che sono multipli di 3

```
int ...
```

Esercizio: numeri perfetti

Un numero intero positivo è chiamato **numero perfetto** se è uguale alla somma dei suoi **divisori propri** (cioè tutti i divisori escluso il numero).

Scrivere un programma che dato un numero intero n , dica se il numero è perfetto.

Esempio:

Sono noti solo 52 numeri perfetti:

I più piccoli sono:

- $6 = 1 + 2 + 3$
- $28 = 1 + 2 + 4 + 7 + 14$
- $496 = 1 + 2 + 4 + 8 + 16 + 31 + 62 + 124 + 248$

Esercizio: numeri perfetti

Scrivere un programma che dato un numero intero n , dica se il numero è perfetto.

```
int i = 1;
int somma = 0;
while (i <= n/2) {
    if (n%i == 0) {
        somma := somma + i;
    }
    i := i + 1;
}
bool perfetto = (somma == n);
```

Esercizi su grammatiche

Alberi di derivazione sintattica

Data la grammatica $G = (\{ S, A, B \}, \{ a, b \}, P)$,
dove P contiene le seguenti produzioni:

$S ::= a A b$

$A ::= a A b \mid B$

$B ::= b B \mid b$

Dire se ciascuna delle seguenti stringhe appartiene al linguaggio $L(S)$:
se appartiene disegnare l'albero di derivazione sintattica,
altrimenti spiegare perché non appartiene

- $aaabbb$
- $aaabbbb$

Induzione strutturale

Data la grammatica $G = (\{ S \}, \{ a, b, c \}, P)$,
dove P contiene le seguenti produzioni:

$S ::= a S b \mid c \mid a c S b$

Dimostrare per induzione strutturale sulla grammatica che, per ogni stringa generata a partire dal simbolo non terminale S , valgono le seguenti proprietà:

- non compare mai una b che stia a sinistra di a o c ;
- il numero di a è uguale al numero di b

Esercizi più complessi

Esercizio: induzione strutturale

Data la grammatica $G = (\{ S \}, \{ (,) \}, P)$,
dove P contiene le seguenti produzioni:

$$S ::= () \mid (S) \mid S S$$

Dimostrare, usando il principio di induzione strutturale sulla grammatica, che ogni stringa generata da S soddisfa la seguente proprietà

- in ogni prefisso della stringa, il numero di (è maggiore o uguale al numero di).

Esercizio: Fibonacci

La successione di Fibonacci è definita induttivamente come:

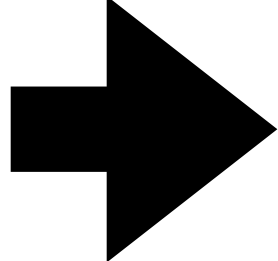
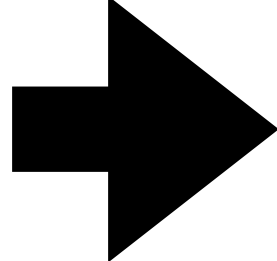
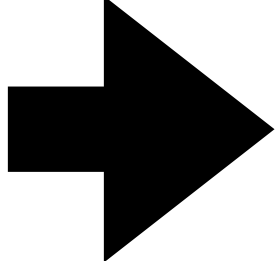
- $F(0) = 0$
- $F(1) = 1$
- $F(n+2) = F(n) + F(n+1)$

Scrivi un programma che dato un numero intero n , calcola l' n -esimo numero della successione di Fibonacci

Esercizio: cifre

Scrivere un programma che, dato un numero intero positivo, calcoli la somma delle sue cifre.

Se il risultato ottenuto ha più di una cifra, ripetere l'operazione sommando nuovamente le cifre, fino a quando non si ottiene un numero a una sola cifra.

Esempio: 183657  $1+8+3+6+5+7 = 28$  $2+8 = 10$  $1+0 = 1$