MiniMao (Modello Astratto Operazionale)

Sintassi e semantica

comandi atomici e composizione sequenziale

Esercizio: scambio di valori interi

Scrivere un programma che scambia i valori di due variabili x e y

```
// (quale stato finale quando x=1 e y=2?)
int temp = y ;
y := x ;
x := temp ;
```

Esercizio per casa: rotazione di valori interi

Scrivere un programma che "ruota" i valori di tre variabili intere x, y e z:
al termine dell'esecuzione x deve contenere il valore iniziale di y, y quello di z
e z quello di x

Esercizio per casa: rotazione di valori interi

Scrivere un programma che "ruota" i valori di tre variabili intere x, y e z:
al termine dell'esecuzione x deve contenere il valore iniziale di y, y quello di z
e z quello di x

```
// basta una variabile temporanea

int temp = x ;

x := y ;

y := z ;

z := temp ;
```

MiniMao (Modello Astratto Operazionale)

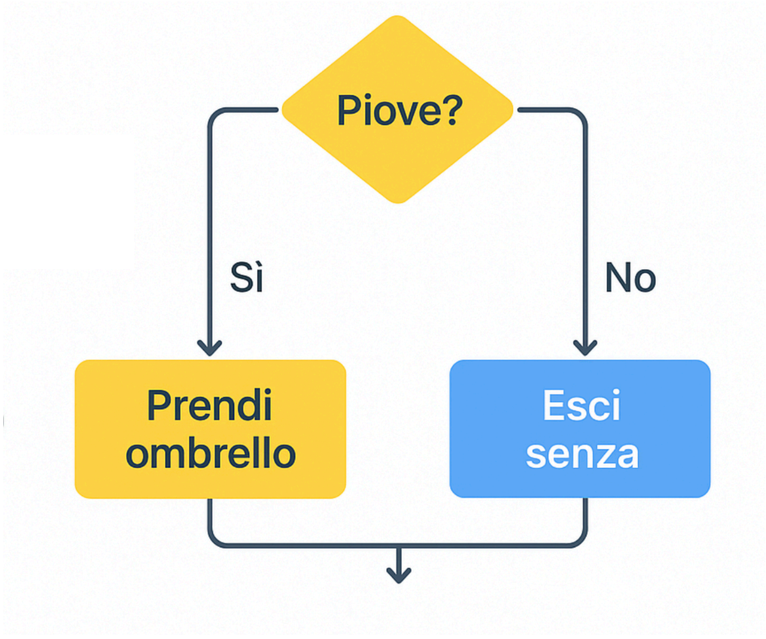
Blocchi, comandi condizionali

Comando condizionale

Comando condizionale

I comandi condizionali servono per prendere decisioni nei programmi

Ad esempio "Se piove, prendo l’ombrello. Altrimenti esco senza."



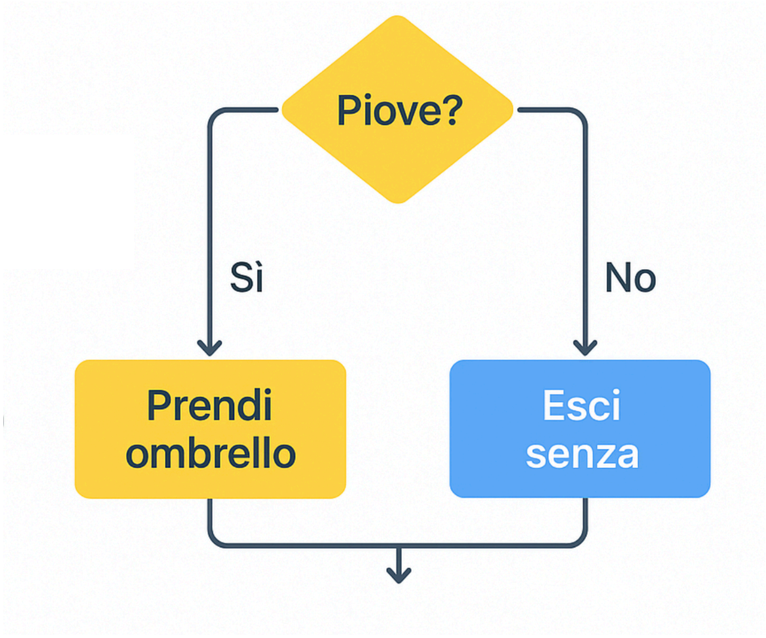
In linguaggio naturale:

- Condizione: *piove?*
- Azione se sì: *prendi ombrello*
- Azione se no: *esci senza*

Comando condizionale in Mao

if (E) { C_1 } else { C_2 }

E è un'espressione booleana: se vera si esegue C_1, altrimenti si esegue C_2



Guardia

```
if ( piove ) {
    ombrello := true;
} else {
    ombrello := false;
}
```

Ramo then

Ramo else

Esercizio: rotazione di valori interi

Scrivere un programma che incrementa o decrementa di 1 il valore di x

a seconda del valore della variabile booleana vincita

```
if (vincita) {
    x := x + 1 ;
} else {
    x := x - 1 ;
}
```

Indentazione e leggibilità

Quale stile preferite?

```
if (vincita) { x := x + 1 ; }
else { x := x - 1 ; }
```

o questo, per ragioni di spazio

```
if (vincita) {
    x := x + 1 ;
} else {
    x := x - 1 ;
}
```

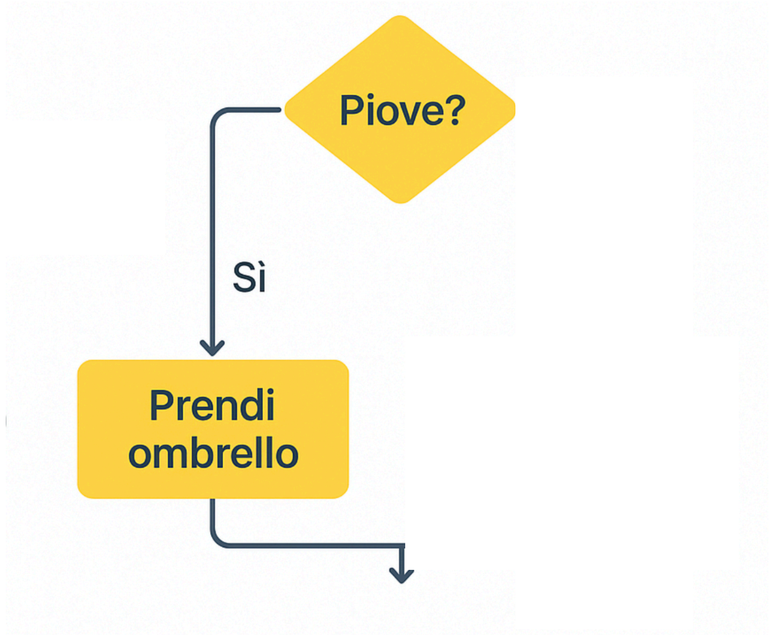
io questo

```
if (vincita)
{
    x := x + 1 ;
}
else
{
    x := x - 1 ;
}
```

```
if (vincita)
    { x := x + 1 ; }
else
    { x := x - 1 ; }
```

Comando If-then

Negli esempi precedenti dovevamo fare azioni in entrambi i casi,
sia quando la guardia era vera che quando era falsa
ma talvolta può essere che dobbiamo fare un azione solo in un caso



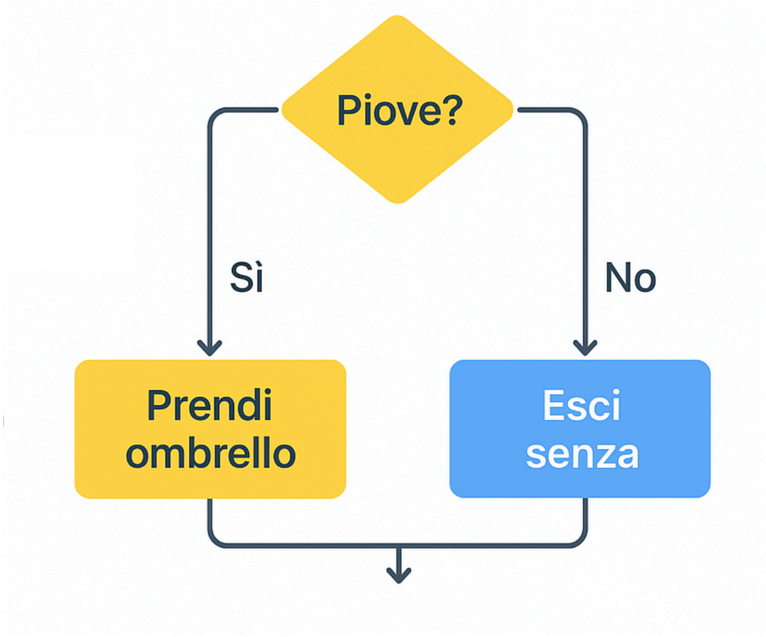
In linguaggio naturale:

- Condizione: *piove?*
- Azione se sì: *prendi ombrello*

Comando if-then in Mao

if (E) { C_1 } else { skip; }

E è un'espressione booleana: se vera si esegue C_1, altrimenti niente



Guardia

```
if ( piove ) {
    ombrello := true;
} else {
    skip;
}
```

Ramo then

Omettiamo ramo else per brevità

Esempio

Scrivere un programma che aggiorna il valore di x con il proprio valore assoluto

```
if ( x < 0 ) {
    x := -x;
} else {
    skip;
}
```

Nel ramo else non faccio niente

```
if ( x < 0 ) {
    x := -x;
}
```

Omettiamo il ramo else

Comando condizionale in Mao

if (E) { C_1 } else { C_2 }

Perché usiamo le parentesi graffe attorno a C_1 e C_2 ? Qual è la conseguenza?

C ::=	skip ;	il comando vacuo
|	T Id = E ;	una dichiarazione (della variabile Id di tipo T col valore di E)
|	Id := E ;	un assegnamento (del valore di E alla variabile Id)
|	C C	composizione sequenziale
|	{ C }	un blocco di comandi
|	if (E) { C } else { C }	un comando condizionale
|	while (E) { C }	un ciclo while

Determina l'ambito delle dichiarazioni!

Blocchi di comandi

(E) contro { C }

```
( 1 + 2 ) * 3
```

```
1 + ( 2 * 3 )
```

Uso le parentesi per
fissare un ordine di valutazione

```
{ x:=1; x:=x+2; } x:=x*3;
```

```
x:=1; { x:=x+2; x:=x*3; }
```

Non fa nessuna differenza:
eseguo le istruzioni nel solito ordine!

```
{ int x=1; x:=x+2; } x:=x*3;
```

una variabile dichiarata all'interno di
un blocco non è visibile all'esterno

```
int x=1; { x:=x+2; x:=x*3; }
```

una variabile è visibile dalla sua dichiarazione in poi,
anche nei sotto-blocchi

Il blocco { C }

Tutte le variabili dichiarate all'interno del blocco sono visibili solo al suo interno (meccanismo utile per riusare variabili locali e impedire conflitti tra nomi)

Lo scope (o ambito di visibilità) di una variabile definisce la porzione di codice nella quale la variabile puó essere dichiarata, letta e modificata

In Mao, le variabili dichiarate all'interno del blocco sono visibili (esclusivamente):

- in tutti i comandi successivi nello stesso blocco
- anche all'interno di tutti gli eventuali blocchi annidati

Esempio: ambito di visibilità (scope)

```
if (vincita) {
    int x = 10 ;
} else {
    int x = 0 ;
}
tot := tot + x;
```

Non può fare riferimento alle variabili dichiarate nei blocchi del condizionale

```
int x ;
if (vincita) {
    x := 10 ;
} else {
    x := 0 ;
}
tot := tot + x;
```

stiamo sempre riferendo la stessa variabile

Shadowing

Per introdurre una nuova variabile all'interno di un blocco possiamo utilizzare lo stesso identificatore di una variabile dichiarata in precedenza: in questo caso, la variabile dichiarata all'interno del blocco nasconde quella dichiarata all'esterno

```
int sconto = 2 ;
if (prezzo > 50) {
    int sconto = 10 ;
    prezzo := prezzo - sconto ;
}
if (prezzo > 200) {
    int sconto = 25 ;
    prezzo := prezzo - sconto ;
}
prezzo := prezzo - sconto ;
```

?

?

?

a quale variabile (tra quelle visibili) si riferisce ciascuna occorrenza di sconto?

sempre a quella più recente!

Shadowing

Per introdurre una nuova variabile all'interno di un blocco possiamo utilizzare lo stesso identificatore di una variabile dichiarata in precedenza: in questo caso, la variabile dichiarata all'interno del blocco nasconde quella dichiarata all'esterno

```
int sconto = 2 ;
if (prezzo > 50) {
    int sconto = 10 ;
    prezzo := prezzo - sconto ;
}
if (prezzo > 200) {
    int sconto = 25 ;
    prezzo := prezzo - sconto ;
}
prezzo := prezzo - sconto ;
```

a quale variabile (tra quelle visibili) si riferisce ciascuna occorrenza di sconto?

sempre a quella più recente!

Esercizio

Scriviamo un programma che assegna a una nuova variabile m il massimo tra i valori delle variabili x, y

```
int m = x ;
if (y > m) {
    m := y ;
}
```

Esercizio

Scriviamo un programma che assegna a una nuova variabile m il massimo tra i valori delle variabili x, y e z

```
int m = x ;
if (y > m) { m := y ; }
else {
    if (z > m) { m := z ; }
}
```

funziona?

Esercizio

Scriviamo un programma che assegna a una nuova variabile m il massimo tra i valori delle variabili x, y e z

```
int m = x ;
if (y > m) { m := y ; }
if (z > m) { m := z ; }
```

meglio!

Esercizio: anno bisestile?

Proviamo a scrivere condizioni (guardie) più complesse:
vogliamo controllare se un anno è bisestile

Un anno è bisestile se:

- è divisibile per 4,
- **ma non** è divisibile per 100,
- **a meno che** non sia anche divisibile per 400.

In altre parole:

- Gli anni come 1996, 2004, 2020 (divisibili per 4 ma non per 100) sono bisestili.
- Gli anni come 1700, 1800, 1900 (divisibili per 100 ma non per 400) **non** sono bisestili.
- Gli anni come 1600, 2000, 2400 (divisibili per 400) **sono** bisestili.

Per il controllo di divisibilità possiamo usare l'operatore % che restituisce il resto della divisione tra interi (es. 8 % 2 = 0, 8 % 3 = 2 e 8 % 5 = 3)

Esercizio: anno bisestile?

Un anno è bisestile se:

- è divisibile per 4,
- **ma non** è divisibile per 100,
- **a meno che** non sia anche divisibile per 400.

Esempi:

- Gli anni come 1996, 2004, 2020 (divisibili per 4 ma non per 100) sono bisestili.
- Gli anni come 1700, 1800, 1900 (divisibili per 100 ma non per 400) **non** sono bisestili.
- Gli anni come 1600, 2000, 2400 (divisibili per 400) **sono** bisestili.

```
bool bis = false ;
if ( ((anno%4 == 0) && (anno%100 != 0))
   || (anno%400 == 0) ) {
    bis := true ;
}
```

possiamo usare una sola istruzione?

Esercizio: anno bisestile?

Un anno è bisestile se:

- è divisibile per 4,
- **ma non** è divisibile per 100,
- **a meno che** non sia anche divisibile per 400.

Esempi:

- Gli anni come 1996, 2004, 2020 (divisibili per 4 ma non per 100) sono bisestili.
- Gli anni come 1700, 1800, 1900 (divisibili per 100 ma non per 400) **non** sono bisestili.
- Gli anni come 1600, 2000, 2400 (divisibili per 400) **sono** bisestili.

```
bool bis =
    ( ((anno%4 == 0) && (anno%100 != 0))
    || (anno%400 == 0) );
```

Esercizio: triangoli

Dati le lunghezze dei tre lati AB, AC e BC di un triangolo, scriviamo un programma per classificarlo come equilatero, isoscele o scaleno (usando tre variabili booleane)

```
bool equi ; // implicitamente = false
bool iso ;
bool sca ;
if ((ab == ac) && (ab == bc)) {
    equi := true ;
} else { // non equilatero
    if ((ab == ac) || (ab == bc) || (ac == bc)) {
        iso := true ;
    } else {
        sca := true ;
    }
}
```

senza ripetere confronti?

Esercizio: triangoli

Dati le lunghezze dei tre lati AB, AC e BC di un triangolo, scriviamo un programma per classificarlo come equilatero, isoscele o scaleno (usando tre variabili booleane)

```
bool equi ; // implicitamente = false
bool iso ;
bool sca ;
if (ab == ac) { // AB e AC uguali: equi o iso
    if (ab == bc) { equi := true ; }
    else { iso := true ; }
} else {  // AB e AC diversi: iso o sca
    if ((ab == bc) || (ac == bc)) { iso := true ; }
    else { sca := true ; }
}
```

Esercizio: triangoli

Dati le lunghezze dei tre lati AB, AC e BC di un triangolo, scriviamo un programma per classificarlo come equilatero, isoscele o scaleno (usando tre variabili booleane)

```
bool equi ; // implicitamente = false
bool iso ;
bool sca ;
if (ab == ac) { // AB e AC uguali: equi o iso
    if (ab == bc) { equi := true ; }
    else { iso := true ; }
} else {  // AB e AC diversi: iso o sca
    if (ab == bc) { iso := true ; }
    else { // AB diverso da AC e da BC: iso o sca
        if (ac == bc) { iso := true ; }
        else { sca := true ; }
    }
}
```

Esercizio per casa: grado alcolico

Supponendo di avere le seguenti 6 variabili booleane:

super, alc, liq, forte, norm e leggero

classificare un vino in base ai gradi alcolici rispetto alla tabella

gradi alcolici g	messaggio
$40 < g$	superalcolico
$20 < g \leq 40$	alcolico
$15 < g \leq 20$	vino liquoroso
$12 < g \leq 15$	vino forte
$10.5 < g \leq 12$	vino normale
$g \leq 10.5$	vino leggero

Esercizio per casa: divisori

Dato un numero n, determinare quanti tra 2,3,5,7 sono suoi divisori
es. 15 ne ha due (3 e 5), 11 nessuno, 70 ne ha tre (2,5,7)

Esercizio per casa: ultimo bisestile

Dato un anno n, determinare qual è stato l'ultimo anno bisestile

Esercizio per casa: prossimo bisestile

Dato un anno n, determinare quale sarà il prossimo anno bisestile

Esercizio per casa: minimo e massimo

Dati tre interi a,b,c determinare il valore minimo e quello massimo

MiniMao: Semantica Operazionale

comandi condizionali e blocchi

Semantica dei comandi (seconda parte)

C ::= ...

Per eseguire il blocco { C } nello stato (ρ , σ) , dobbiamo:

- eseguire il comando C nello stato (ρ , σ) ottenendo (ρ_1 , σ_1)
- restituire (ρ , σ_1) come stato finale in modo da ripristinare l'ambiente iniziale ρ e preservare la memoria modificata σ_1

| { C }

$$\frac{\langle C , \rho , \sigma \rangle \rightarrow \langle \rho_1 , \sigma_1 \rangle}{\langle \{ C \} , \rho , \sigma \rangle \rightarrow \langle \rho , \sigma_1 \rangle}$$

| if (E) { C_1 } else { C_2 }

Diversamente da tutti i casi visti, per il comando condizionali abbiamo bisogno di due regole:

- una per il caso in cui la guardia E è vera (ramo then)
- una per il caso in cui la guardia E è falsa (ramo else)

| ...

Semantica dei comandi (seconda parte)

C ::= ...

Per eseguire il condizionale nello stato (ρ , σ) , dobbiamo:

- valutiamo la guardia E nello stato (ρ , σ) ottenendo v
- se v è true lo stato finale sarà ottenuto eseguendo C_1 in (ρ , σ)
- se v è false lo stato finale sarà ottenuto eseguendo C_2 in (ρ , σ)
- trattandosi di blocchi, si ripristina sempre l'ambiente iniziale ρ

| { C }

| if (E) { C_1 } else { C_2 }

$$\frac{\langle E , \rho , \sigma \rangle \Downarrow true \quad \langle C_1 , \rho , \sigma \rangle \rightarrow \langle \rho_1 , \sigma_1 \rangle}{\langle \text{if (E) \{ } C_1 \text{ \} else \{ } C_2 \text{ \} } , \rho , \sigma \rangle \rightarrow \langle \rho , \sigma_1 \rangle}$$

$$\frac{\langle E , \rho , \sigma \rangle \Downarrow false \quad \langle C_2 , \rho , \sigma \rangle \rightarrow \langle \rho_2 , \sigma_2 \rangle}{\langle \text{if (E) \{ } C_1 \text{ \} else \{ } C_2 \text{ \} } , \rho , \sigma \rangle \rightarrow \langle \rho , \sigma_2 \rangle}$$

| ...

Semantica operazionale dei comandi (prima parte)

$$\frac{}{\langle\ \mathsf{skip;}\ ,\ \rho\ ,\ \sigma\ \rangle \rightarrow \langle\ \rho\ ,\ \sigma\ \rangle} \qquad \frac{\langle\ \mathsf{C}_1\ ,\ \rho\ ,\ \sigma\ \rangle \rightarrow \langle\ \rho_1\ ,\ \sigma_1\ \rangle \quad \langle\ \mathsf{C}_2\ ,\ \rho_1\ ,\ \sigma_1\ \rangle \rightarrow \langle\ \rho_2\ ,\ \sigma_2\ \rangle}{\langle\ \mathsf{C}_1\ \mathsf{C}_2\ ,\ \rho\ ,\ \sigma\ \rangle \rightarrow \langle\ \rho_2\ ,\ \sigma_2\ \rangle}$$

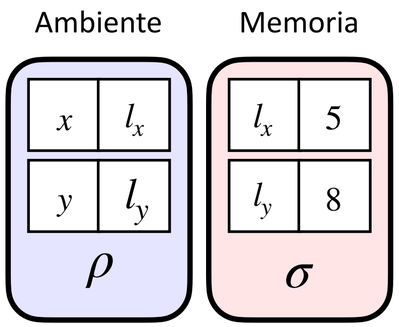
$$\frac{\langle\ \mathsf{E}\ ,\ \rho\ ,\ \sigma\ \rangle \Downarrow v \quad l \notin \sigma}{\langle\ \mathsf{T\ Id\ =\ E;}\ ,\ \rho\ ,\ \sigma\ \rangle \rightarrow \langle \rho[\mathsf{Id} \mapsto l]\ ,\ \sigma[l \mapsto v]\ \rangle} \qquad \frac{\langle\ \mathsf{E}\ ,\ \rho\ ,\ \sigma\ \rangle \Downarrow v \quad \rho(\mathsf{Id}) = l}{\langle\ \mathsf{Id\ :=\ E;}\ ,\ \rho\ ,\ \sigma\ \rangle \rightarrow \langle \rho\ ,\ \sigma[l \mapsto v]\ \rangle}$$

$$\frac{\langle\ \mathsf{C}\ ,\ \rho\ ,\ \sigma\ \rangle \rightarrow \langle\ \rho_1\ ,\ \sigma_1\ \rangle}{\langle\ \mathsf{\{\ C\ \}}\ ,\ \rho\ ,\ \sigma\ \rangle \rightarrow \langle\ \rho\ ,\ \sigma_1\ \rangle}$$

$$\frac{\langle\ \mathsf{E}\ ,\ \rho\ ,\ \sigma\ \rangle \Downarrow \mathit{true} \quad \langle\ \mathsf{C}_1\ ,\ \rho\ ,\ \sigma\ \rangle \rightarrow \langle\ \rho_1\ ,\ \sigma_1\ \rangle}{\langle\ \mathsf{if\ (\ E\ )\ \{\ C_1\ \}\ else\ \{\ C_2\ \}}\ ,\ \rho\ ,\ \sigma\ \rangle \rightarrow \langle\ \rho\ ,\ \sigma_1\ \rangle}$$

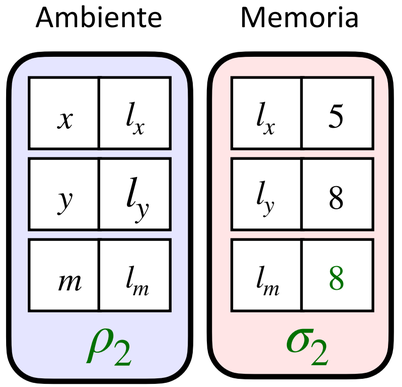
$$\frac{\langle\ \mathsf{E}\ ,\ \rho\ ,\ \sigma\ \rangle \Downarrow \mathit{false} \quad \langle\ \mathsf{C}_2\ ,\ \rho\ ,\ \sigma\ \rangle \rightarrow \langle\ \rho_2\ ,\ \sigma_2\ \rangle}{\langle\ \mathsf{if\ (\ E\ )\ \{\ C_1\ \}\ else\ \{\ C_2\ \}}\ ,\ \rho\ ,\ \sigma\ \rangle \rightarrow \langle\ \rho\ ,\ \sigma_2\ \rangle}$$

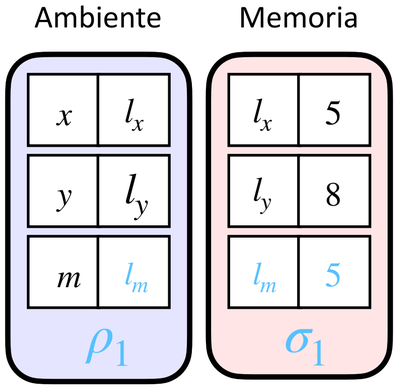
Esempi di esecuzione di comandi



C_1 = `int m=x;`

C_2 = `if (y>m) { m:=y;}`





$\langle C_1\ C_2 , \rho , \sigma \rangle \rightarrow \langle \rho_2 , \sigma_2 \rangle$

$\langle$ int $m = x;$, ρ , σ $\rangle \rightarrow \langle \rho_1 , \sigma_1 \rangle$

$$\frac{\langle E , \rho , \sigma \rangle \Downarrow v \quad l \notin \sigma}{\langle T\ Id\ = E; , \rho , \sigma \rangle \rightarrow \langle \rho[Id \mapsto l] , \sigma[l \mapsto v] \rangle}$$

$\langle x , \rho , \sigma \rangle \Downarrow 5 \quad l_m \notin \sigma \quad \rho_1 = \rho[m \mapsto l_m] \quad \sigma_1 = \sigma[l_m \mapsto 5]$

$\langle$ if $(y>m)$ $\{ m:=y;\}$, ρ_1 , σ_1 $\rangle \rightarrow \langle \rho_1 , \sigma_2 \rangle \quad \rho_2 = \rho_1$

$$\frac{\langle E , \rho , \sigma \rangle \Downarrow true \quad \langle C_1 , \rho , \sigma \rangle \rightarrow \langle \rho_1 , \sigma_1 \rangle}{\langle \text{if } (E) \{ C_1 \} \text{ else } \{ C_2 \} , \rho , \sigma \rangle \rightarrow \langle \rho , \sigma_1 \rangle}$$

$\langle y > m; , \rho_1 , \sigma_1 \rangle \Downarrow true$ // omettiamo i dettagli!

$\langle m := y; , \rho_1 , \sigma_1 \rangle \rightarrow \langle \rho_1 , \sigma_2 \rangle$

$$\frac{\langle E , \rho , \sigma \rangle \Downarrow v \quad \rho(Id) = l}{\langle Id\ := E; , \rho , \sigma \rangle \rightarrow \langle \rho , \sigma[l \mapsto v] \rangle}$$

$\langle y , \rho_1 , \sigma_1 \rangle \Downarrow 8 \quad \sigma_2 = \sigma_1[l_m \mapsto 8]$