

Fondamenti di Programmazione (A)

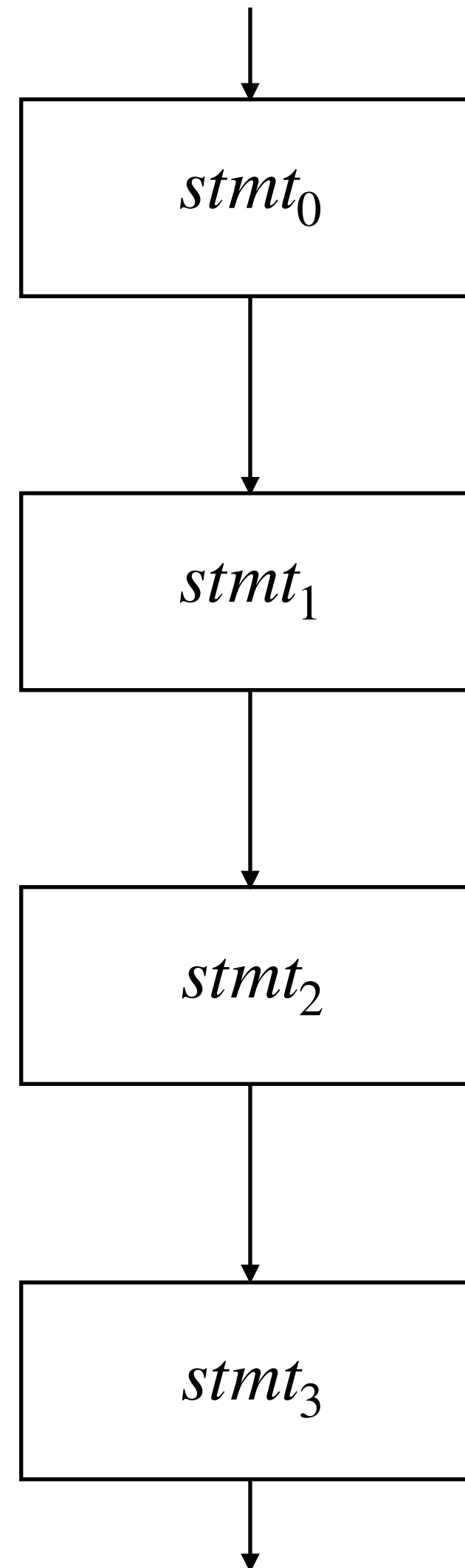
7 - Costrutti di controllo del flusso - Statement `if`

Vincenzo Arceri - Università degli Studi di Parma - vincenzo.arceri@unipr.it

Puntate precedenti

- Assegnamenti e operatori di assegnamento
- Espressioni
- Tipo e valore di un'espressione
- Side-effects

Programmi sequenziali



Finora: Il *flusso*
dell'esecuzione non
cambia e ogni statement
viene eseguito uno
dietro l'altro
(in sequenza)

Oggi: introdurremo
statement per cambiare
il flusso d'esecuzione del
programma

Blocco

- Il blocco è una regione testuale del programma utilizzata per raggruppare più comandi

```
{  
    x = 1;  
    y = 3;  
    int z = 1;  
    z = z * x + y;  
}
```

Blocco

- Il blocco è una regione testuale del programma utilizzata per raggruppare più comandi

```
{  
    x = 1;  
    y = 3;  
    int z = 1;  
    z = z * x + y;  
}
```

- I blocchi possono essere innestati

Blocco

- Il blocco è una regione testuale del programma utilizzata per raggruppare più comandi

```
{  
    x = 1;  
    y = 3;  
    int z = 1;  
    z = z * x + y;  
}
```

- I blocchi possono essere innestati

```
{  
    x = 1;  
    y = 3;  
    {  
        int z = 1;  
        z = z * x + y;  
    }  
}
```

Blocco

- Il blocco è una regione testuale del programma utilizzata per raggruppare più comandi

```
{  
    x = 1;  
    y = 3;  
    int z = 1;  
    z = z * x + y;  
}
```

- I blocchi possono essere innestati

```
{  
    x = 1;  
    y = 3;  
    {  
        int z = 1;  
        z = z * x + y;  
    }  
}
```

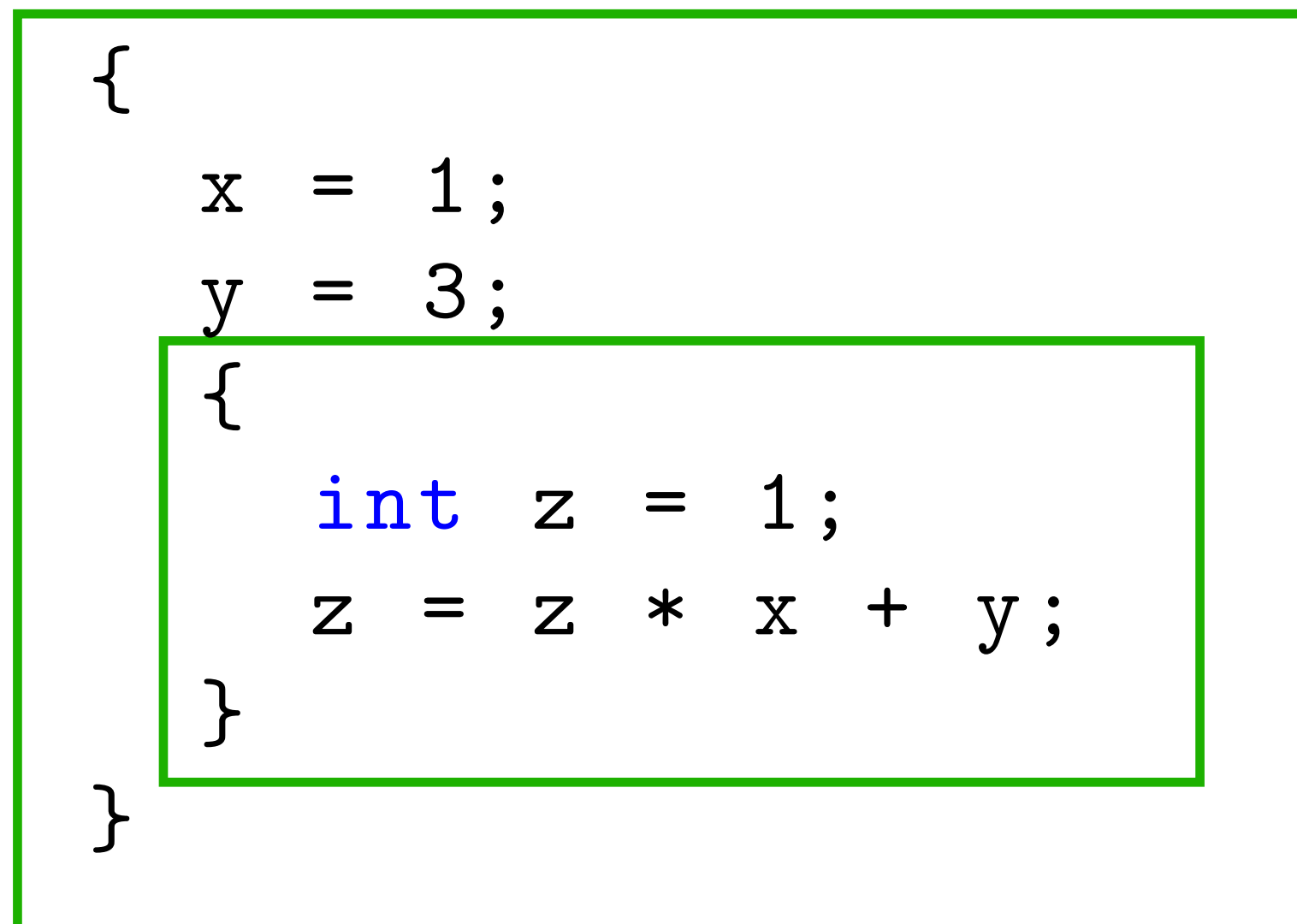
- I blocchi possono essere vuoti

Blocco

- Il blocco è una regione testuale del programma utilizzata per raggruppare più comandi

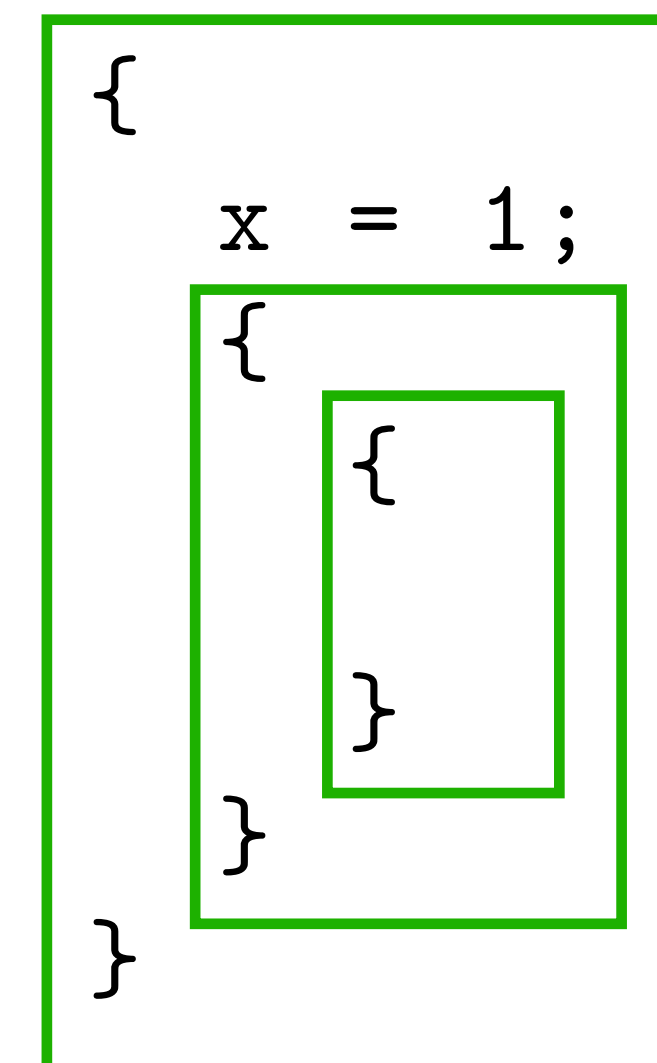
```
{  
    x = 1;  
    y = 3;  
    int z = 1;  
    z = z * x + y;  
}
```

- I blocchi possono essere innestati



```
{  
    x = 1;  
    y = 3;  
    {  
        int z = 1;  
        z = z * x + y;  
    }  
}
```

- I blocchi possono essere vuoti



```
{  
    x = 1;  
    {  
        {  
        }  
    }  
}
```


Statement if – else

Statement if – else

- **Problema:** dato in input il voto di uno studente, stampare a video se lo studente ha passato o meno l'esame

Statement if – else

- **Problema:** dato in input il voto di uno studente, stampare a video se lo studente ha passato o meno l'esame

```
if (exp)  
    stmt1  
else  
    stmt2
```

Statement if – else

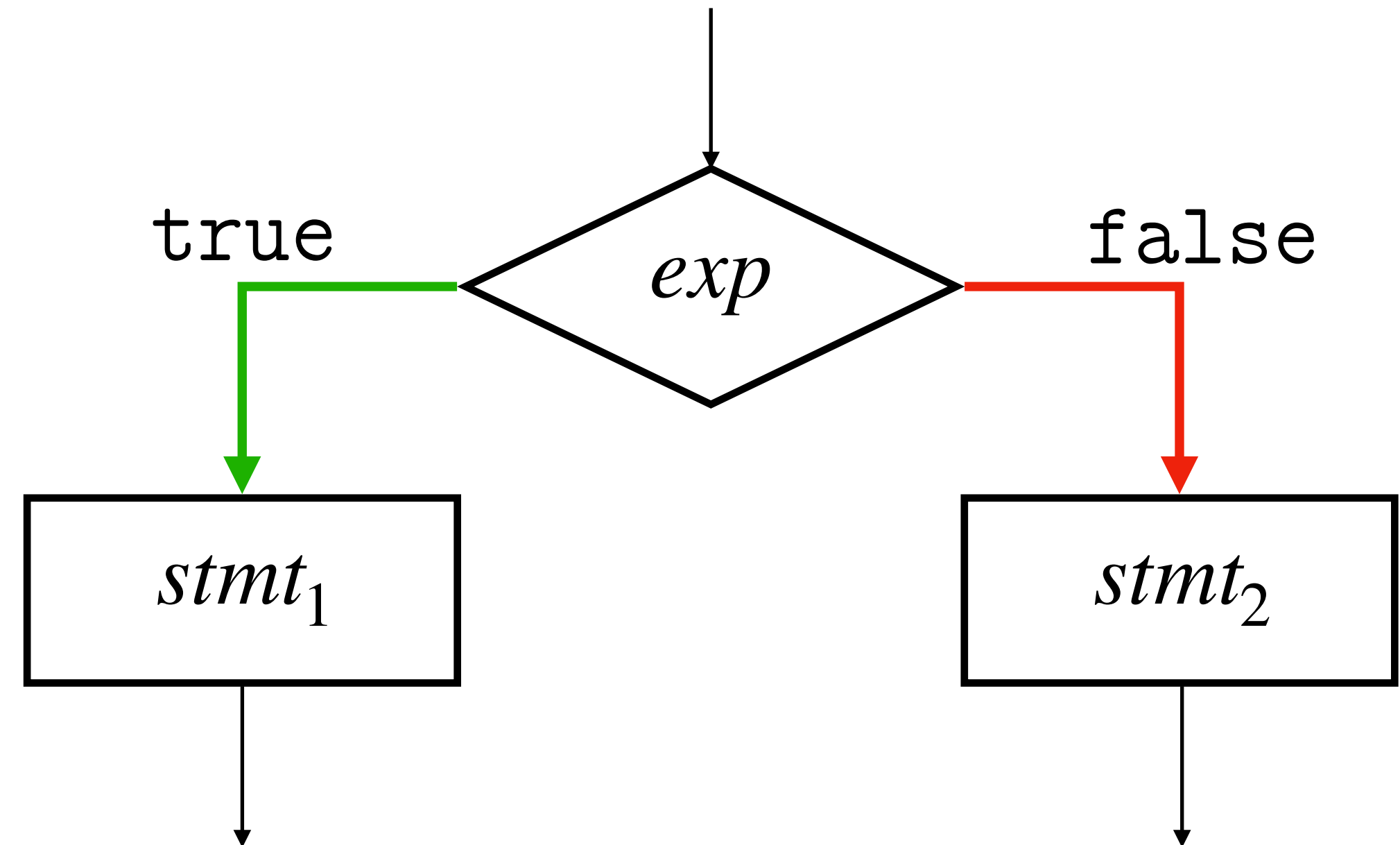
- **Problema:** dato in input il voto di uno studente, stampare a video se lo studente ha passato o meno l'esame

```
if (exp)  
    stmt1  
else  
    stmt2
```

- Informalmente: se *exp* è vera, allora esegui *stmt*₁, altrimenti esegui *stmt*₂

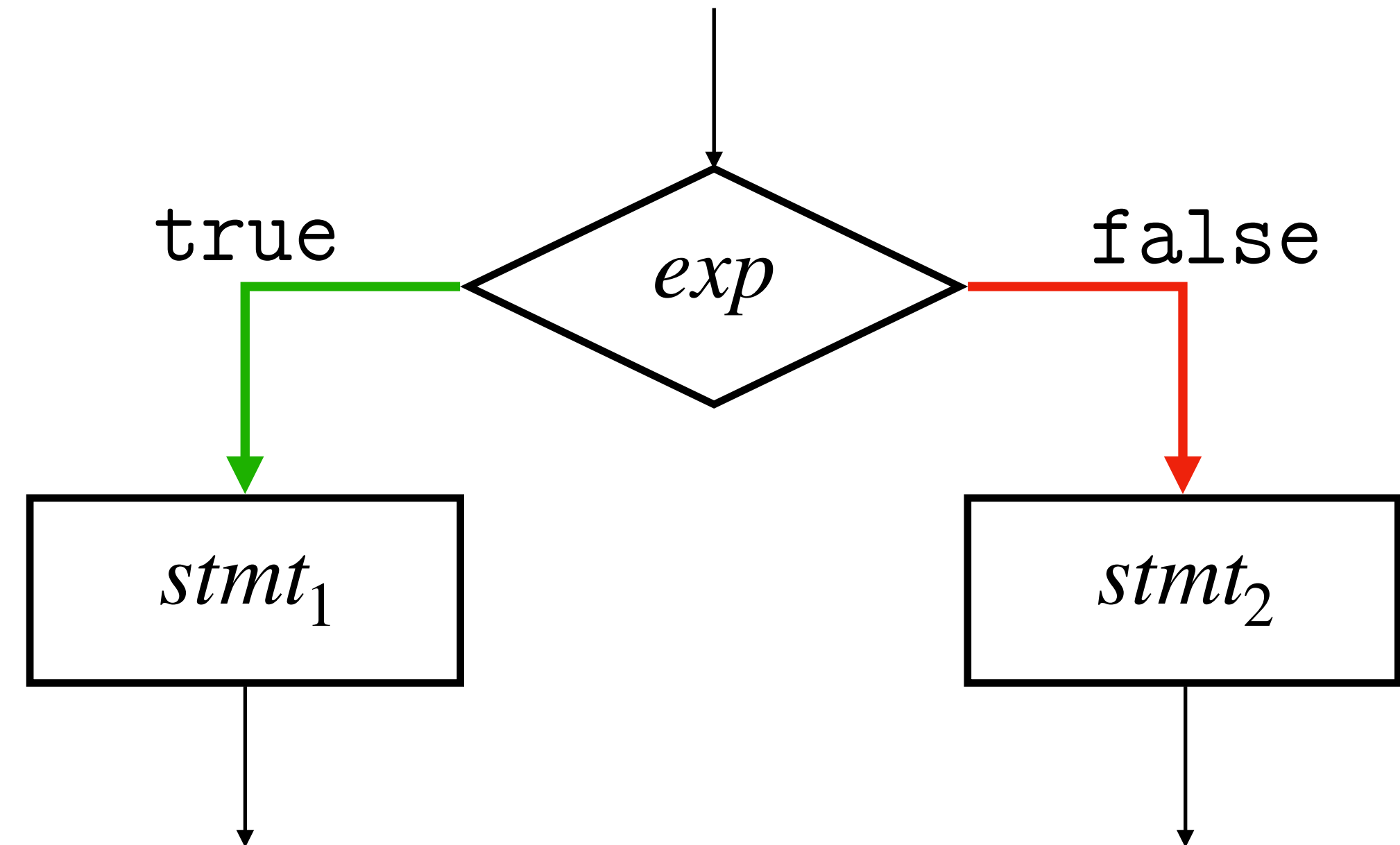
Statement if – else

```
if (exp)  
    stmt1  
else  
    stmt2
```



Statement if – else

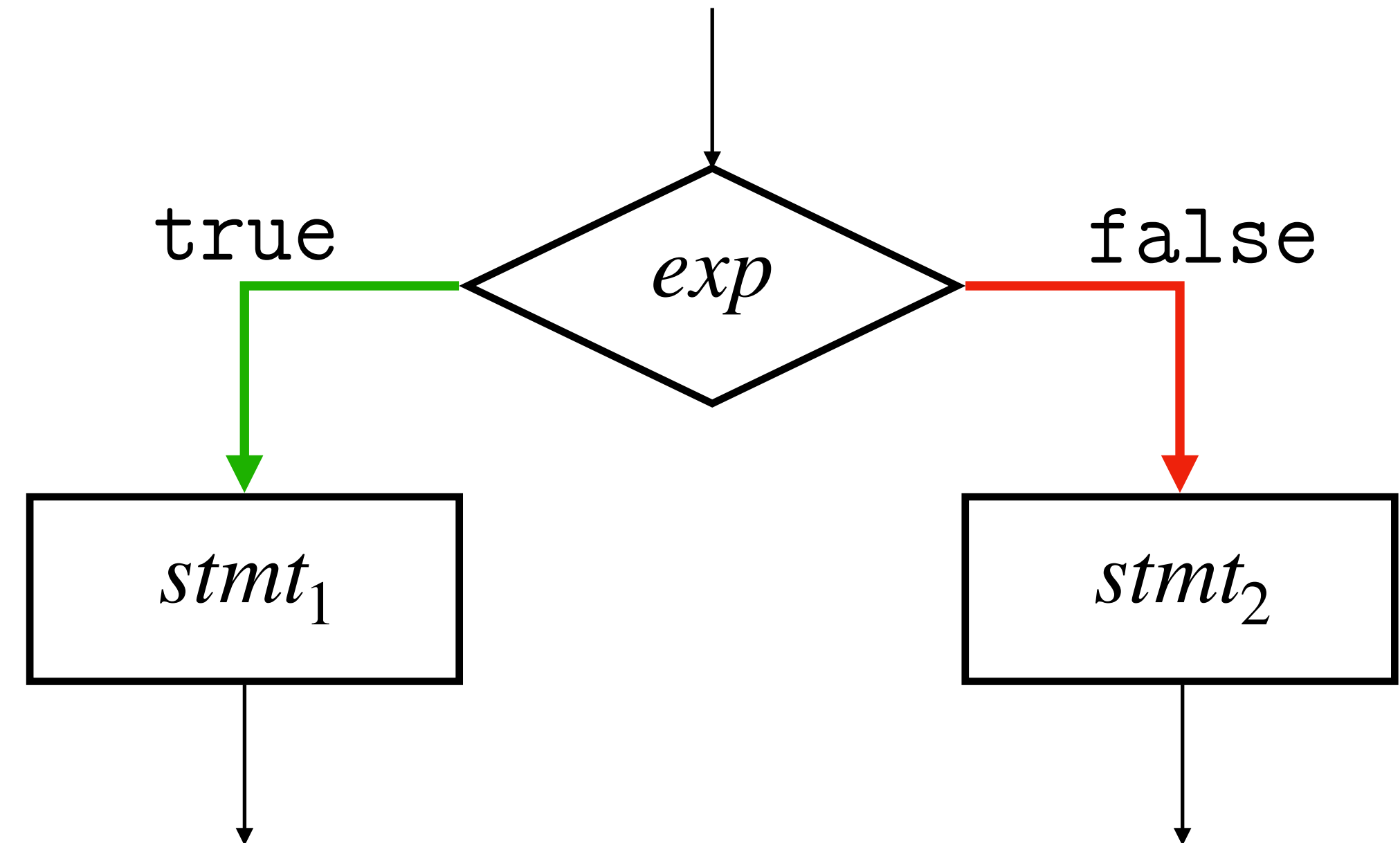
```
if (exp)  
    stmt1  
else  
    stmt2
```



- Biforcazione dell'esecuzione
- Eseguo *stmt*₁ oppure *stmt*₂, a seconda del valore booleano di *exp*

Statement if – else

```
if (exp)  
    stmt1  
else  
    stmt2
```

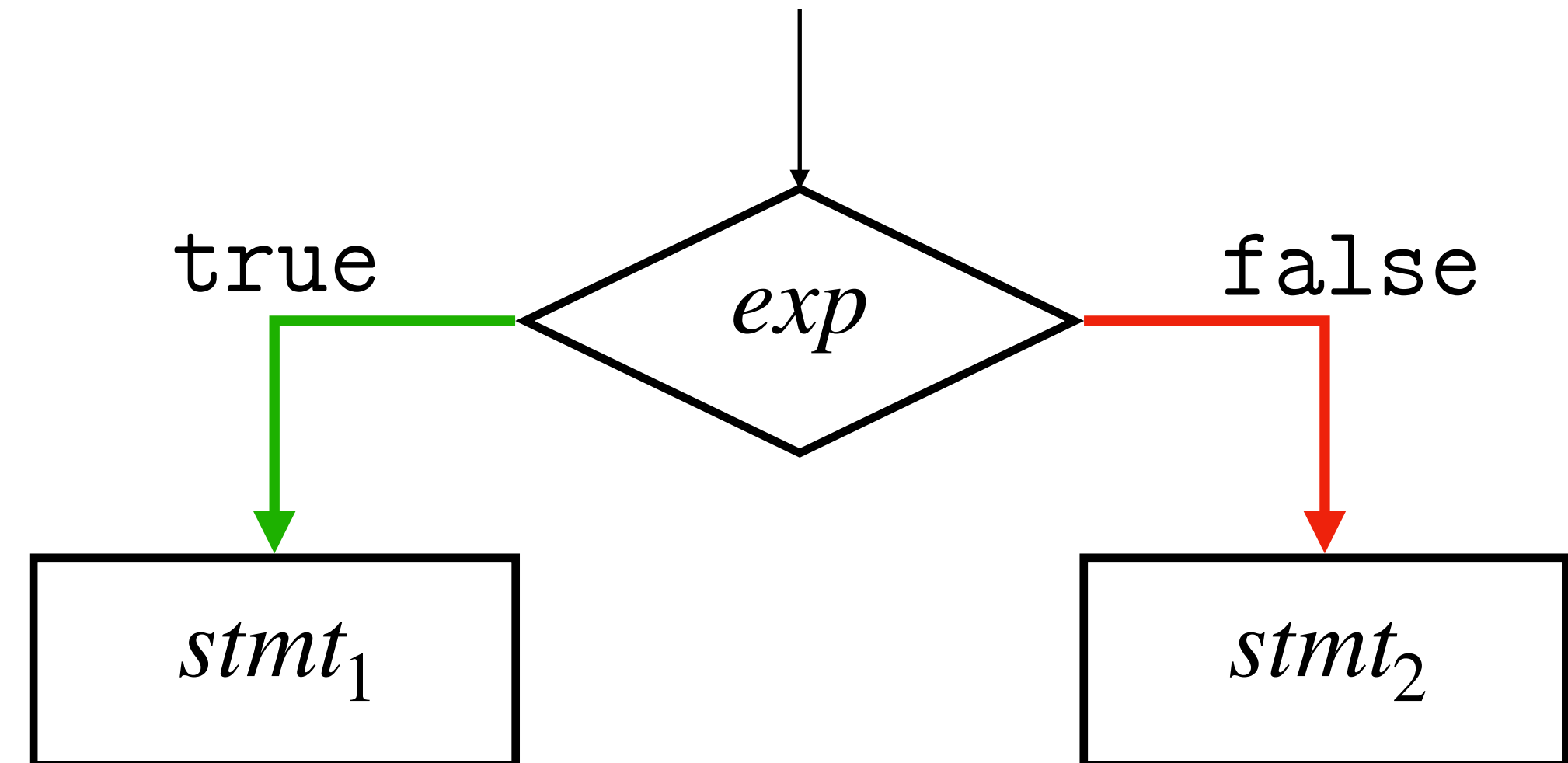


- Biforcazione dell'esecuzione
- Eseguo *stmt*₁ oppure *stmt*₂, a seconda del valore booleano di *exp*

Come prosegue
l'esecuzione dopo aver
eseguito *stmt*₁ oppure
*stmt*₂?

Statement if – else

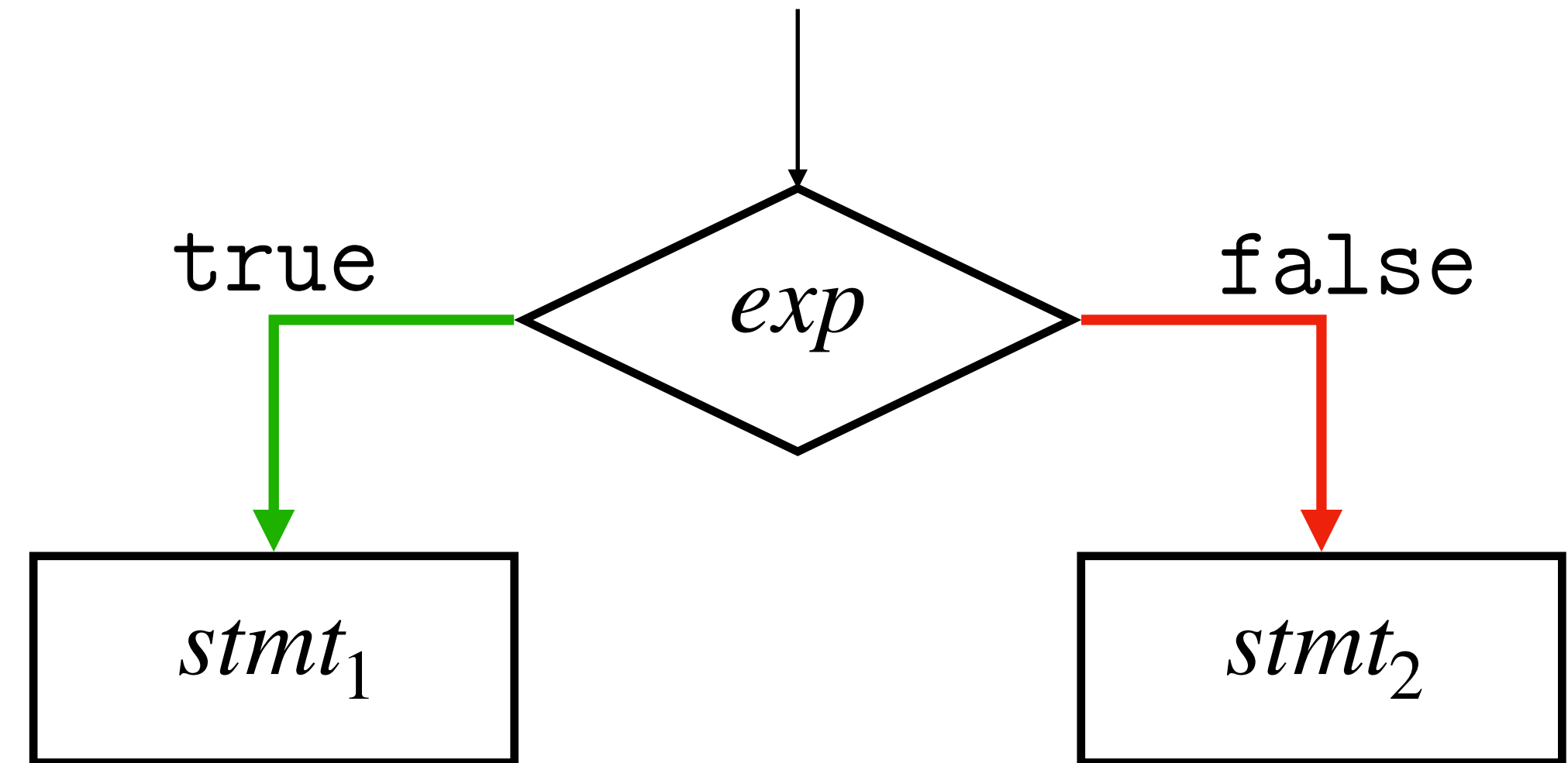
```
if (exp)  
    stmt1  
else  
    stmt2
```



- Biforcazione dell'esecuzione
- Eseguo *stmt*₁ oppure *stmt*₂, a seconda del valore booleani di *exp*

Statement if – else

```
if (exp)  
    stmt1  
else  
    stmt2  
[next]
```



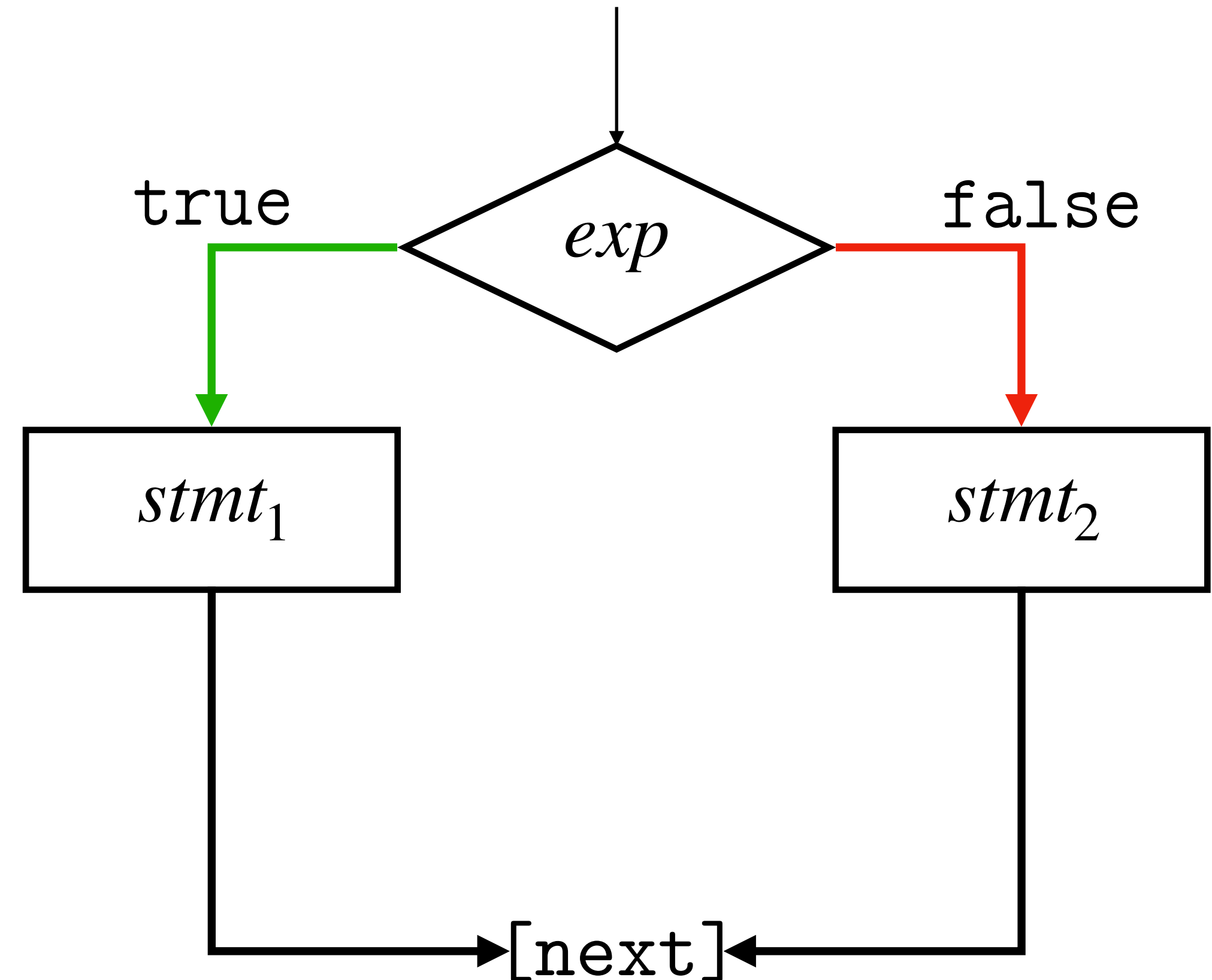
- Biforcazione dell'esecuzione
- Eseguo *stmt*₁ oppure *stmt*₂, a seconda del valore booleani di *exp*

Statement if – else

```
if (exp)  
    stmt1  
else  
    stmt2  
[next]
```

- Biforcazione dell'esecuzione

- Eseguo *stmt*₁ oppure *stmt*₂, a seconda del valore booleani di *exp*



Statement `if – else`

Statement if – else

- $stmt_1$ oppure $stmt_2$ sono statement qualsiasi, quindi anche **blocchi**
- Utile nel caso in cui vogliamo eseguire più statement

Statement if – else

- $stmt_1$ oppure $stmt_2$ sono statement qualsiasi, quindi anche **blocchi**
- Utile nel caso in cui vogliamo eseguire più statement

```
if (exp) {  
    stmt1  
    stmt2  
    ...  
} else {  
    stmt3  
    stmt4  
    ...  
}
```

Statement if – else

Statement if – else

- $stmt_1$ oppure $stmt_2$ sono statement qualsiasi, quindi anche **altri if-else-statement**

Statement if – else

- $stmt_1$ oppure $stmt_2$ sono statement qualsiasi, quindi anche **altri if-else-statement**

```
if ( $exp_1$ )  
     $stmt_1$   
else  
    if ( $exp_2$ )  
         $stmt_2$   
    else  
        if ( $exp_3$ )  
             $stmt_3$   
        else  
             $stmt_4$ 
```

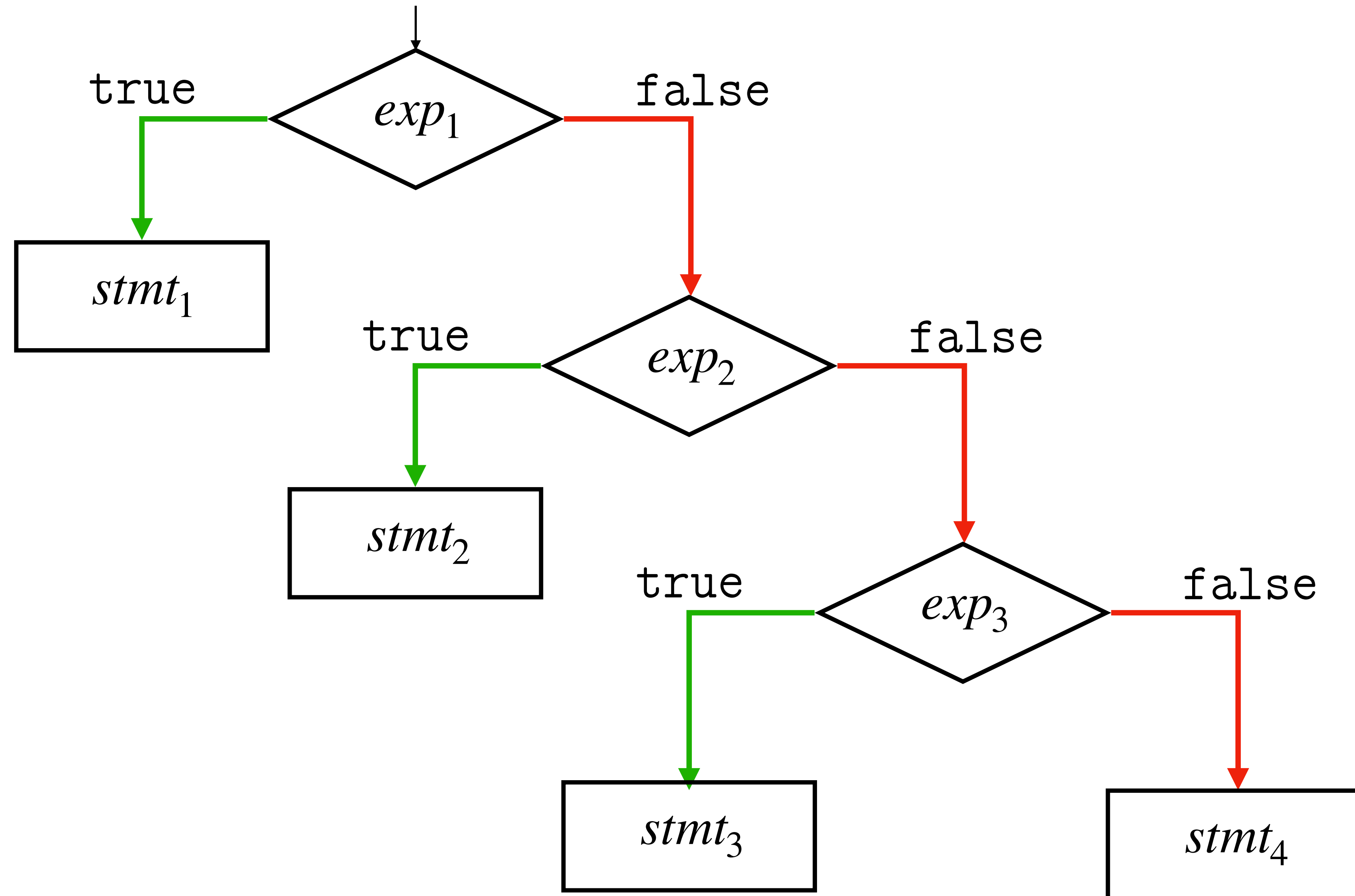

Statement if – else

Statement if – else

```
if ( $exp_1$ )  
     $stmt_1$   
else  
    if ( $exp_2$ )  
         $stmt_2$   
    else  
        if ( $exp_3$ )  
             $stmt_3$   
        else  
             $stmt_4$ 
```

Statement if – else

```
if ( $exp_1$ )  
   $stmt_1$   
else  
  if ( $exp_2$ )  
     $stmt_2$   
  else  
    if ( $exp_3$ )  
       $stmt_3$   
    else  
       $stmt_4$ 
```



Statement `if` – `else`

Esempio

Statement `if` – `else`

Esempio

- **Problema:** dati in input tre valori interi, stampare a video il massimo dei tre

Statement if – else

Esempio

- **Problema:** data in input la lunghezza dei tre lati di un triangolo, si determini se il triangolo è equilatero, isoscele o scaleno

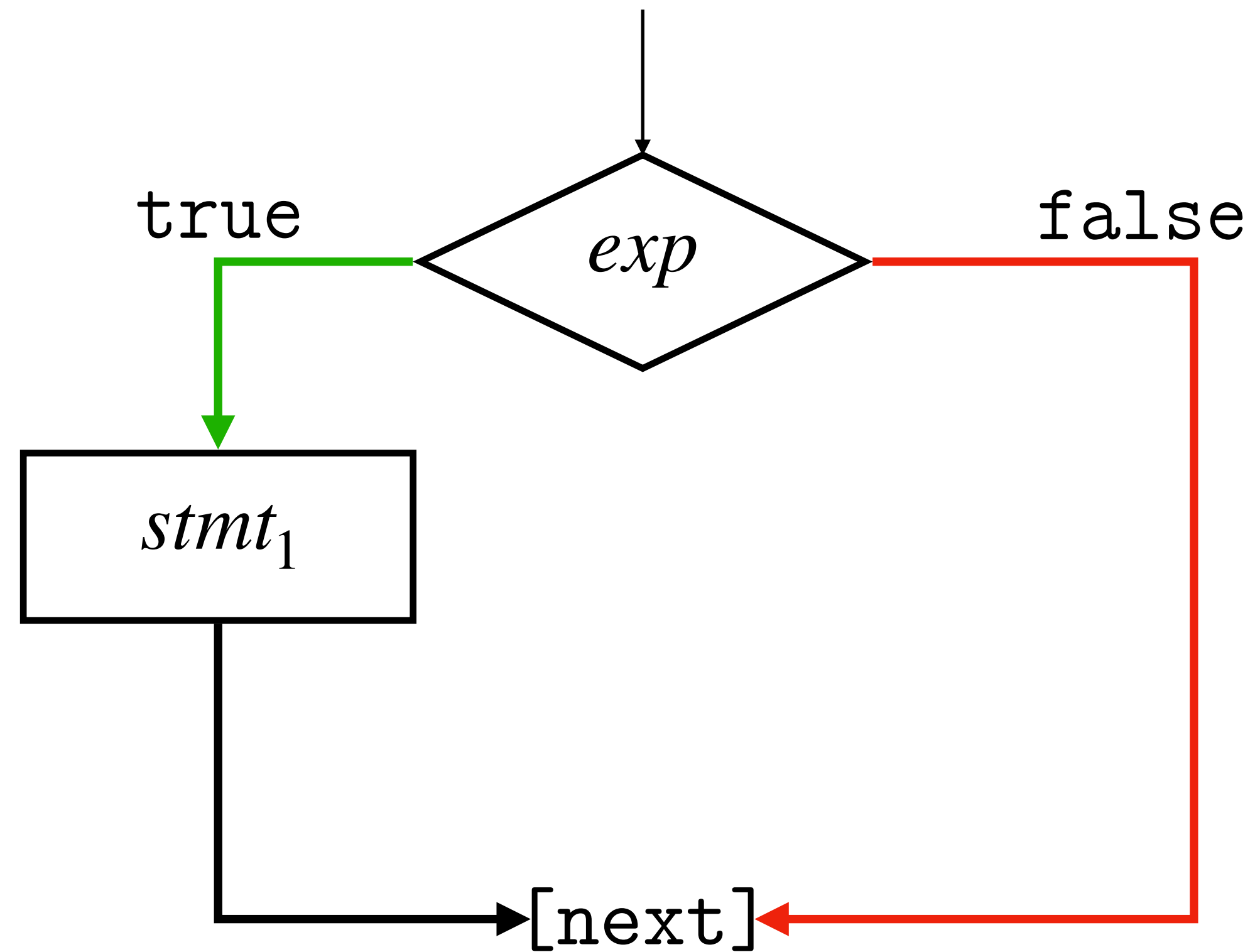
Statement if – else

Esempio

- **Problema:** dato in input il numero di bottiglie di vino che si vuole acquistare, stampare il costo totale (prezzo unitario: 15€). Se le bottiglie di vino sono più di 50, applicare uno sconto del 18%

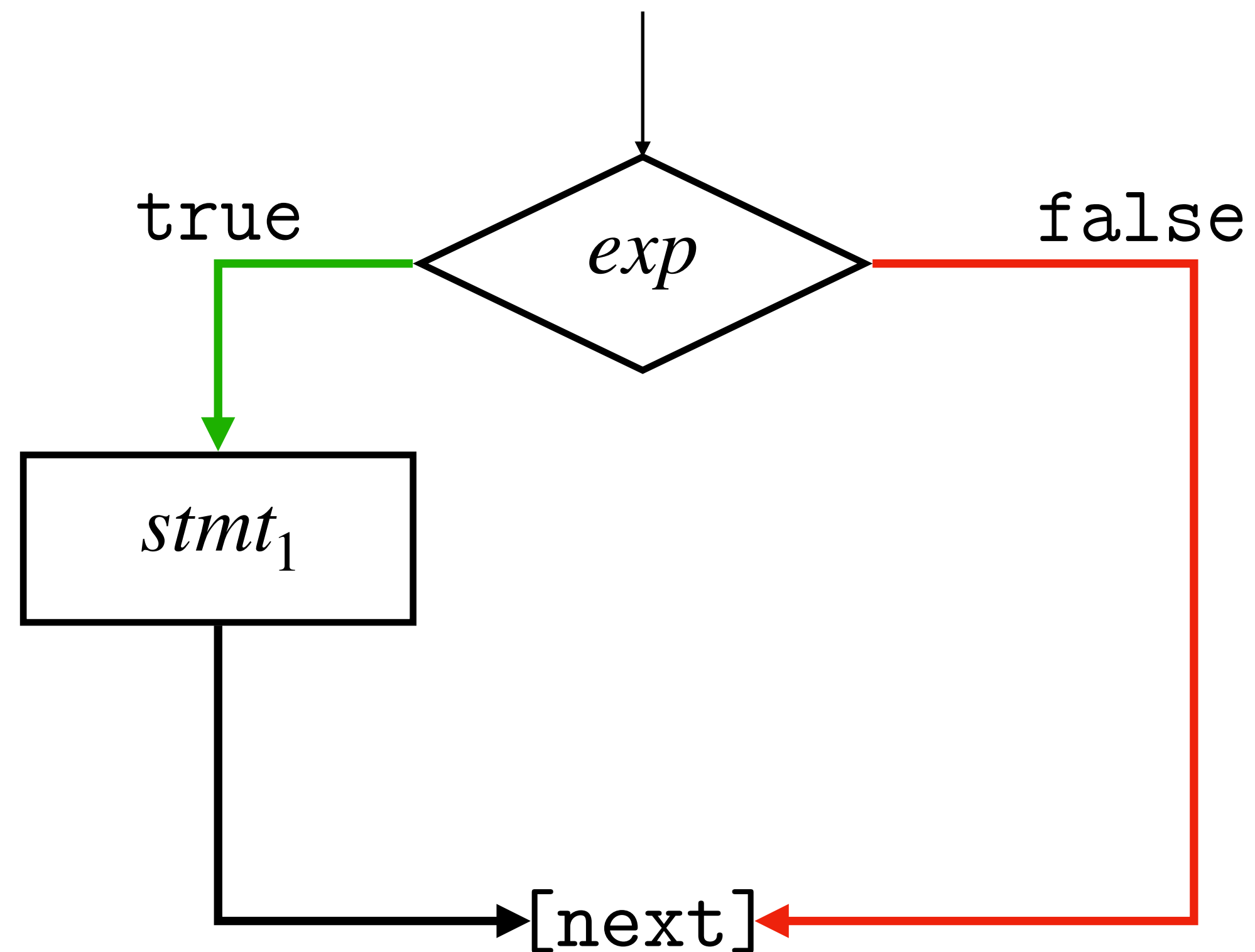
Statement if (senza else)

```
if (exp)  
    stmt  
[next]
```



Statement if (senza else)

```
if (exp)  
    stmt  
[next]
```



(equivalente)

```
if (exp)  
    stmt  
else {  
}  
[next]
```

Statement if – else

Esempio

- **Problema:** dato in input un voto da 0 a 30, stampare a video un giudizio
 - $0 \leq \text{voto} \leq 10$: gravemente insufficiente
 - $10 < \text{voto} \leq 17$: insufficiente
 - $18 \leq \text{voto} \leq 24$: sufficiente
 - $24 < \text{voto} \leq 27$: buono
 - $\text{voto} > 27$: ottimo