

Diagrammi di Flusso

Federico Bolelli
federico.bolelli@unimore.it

Diagramma di Flusso

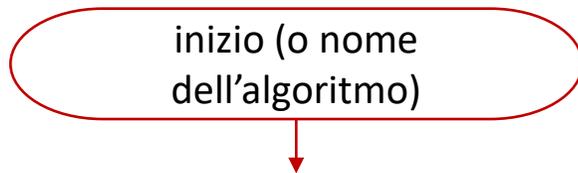
- Il diagramma di flusso (flow chart) è una rappresentazione grafica delle operazioni da eseguire per l'esecuzione di un algoritmo;
- Esso consente di descrivere, tramite un linguaggio di modellazione grafico:
 - le operazioni da compiere, rappresentate mediante sagome convenzionali;
 - la sequenza nella quale devono essere compiute, rappresentata con frecce di collegamento;

Diagramma di Flusso

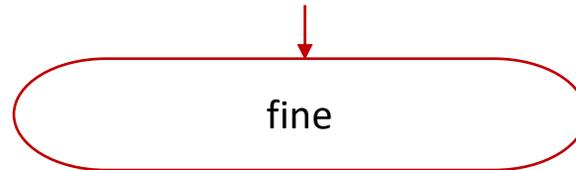
- Il diagramma di flusso è caratterizzato da una lettura sequenziale:
 1. si parte dal blocco iniziale;
 2. si segue la freccia in uscita;
 3. si giunge al blocco successivo e si effettua l'operazione descritta nel blocco;
 4. si procede iterando i passi 2 e 3 fino a giungere al blocco finale.
- Tra le operazioni si distinguono:
 - Azione: una attività o un'elaborazione da svolgere
 - Test: indica due direzioni in base a un fattore di decisione (vero o falso)
 - Ingresso/Uscita: comporta l'immissione di informazioni dall'esterno oppure l'invio di informazioni verso l'esterno

Diagramma di Flusso

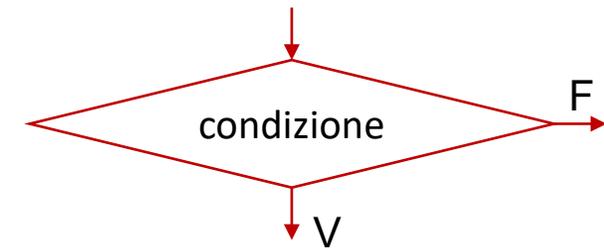
Blocco Iniziale



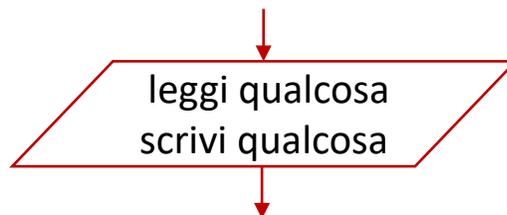
Blocco Finale



Blocco Decisionale o di Test o di Selezione



Blocco di Input/Output



Blocco di Elaborazione



Diagramma di Flusso

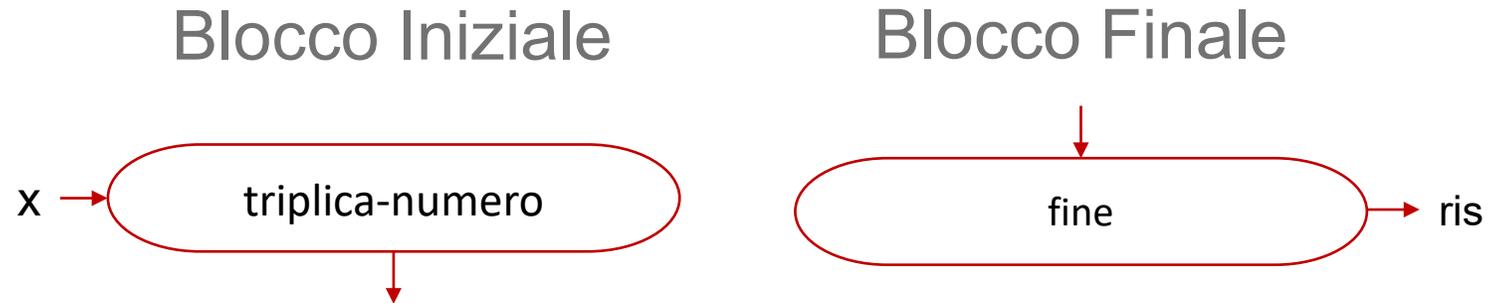
- Una combinazione di blocchi elementari descrive un algoritmo se:
 - viene usato un numero finito di blocchi;
 - lo schema inizia con un blocco iniziale e termina con un blocco finale;
 - ogni blocco soddisfa le condizioni di validità;
- Condizioni di validità:
 - Il blocco assegnamento (lettura e scrittura) e il blocco elaborazione hanno una sola freccia entrante e una sola freccia uscente;
 - Il blocco di controllo ha una freccia entrante e due frecce uscenti etichettate con V (Vero) e F (Falso);
 - Ogni freccia deve entrare in un blocco;
 - Dal blocco iniziale deve essere possibile raggiungere ogni blocco;
 - Da ogni blocco dev'essere possibile raggiungere il blocco finale;

Diagramma di Flusso

- A volte vi è la necessità di modellare algoritmi che presuppongono di conoscere determinati parametri (*input*) o che forniscono un risultato (*output*).
- L'algoritmo *triplica-numero*, ad esempio, deve ricevere in ingresso il numero di cui calcolare il triplo.
- Per poter far riferimento a questo numero arbitrario, come in matematica, abbiamo la necessità di dargli un **nome**.
- Per questo motivo, nell'ambito del corso estendiamo i blocchi di *inizio* e di *fine* aggiungendo la possibilità di specificare parametri di *input* e *output*.

Blocchi Inizio e Fine

- La rappresentazione che utilizzeremo è la seguente:



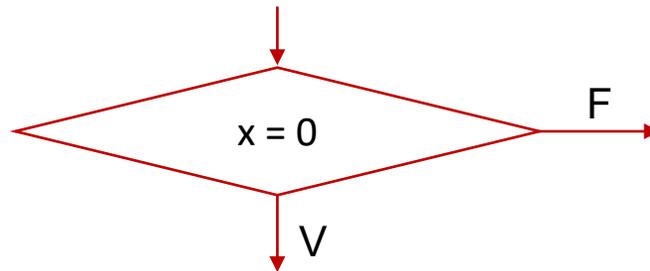
Blocchi Inizio e Fine

- Se ci sono più parametri li separiamo con virgole:



Blocco Condizione/Selezione/Test

- A seconda delle capacità del sistema, è possibile utilizzare condizioni più o meno complesse.
- Un sistema potrebbe essere in grado di fare verifiche solo del tipo $a > b$, $a < b$ o $a = b$, oppure potrebbe consentire anche verifiche più complesse come $4 < a < 10$.



Blocco di Elaborazione

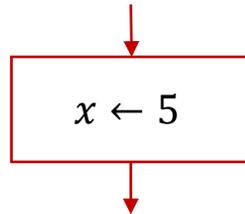
- Un discorso analogo può essere fatto per le operazioni che è possibile eseguire nei blocchi di elaborazione.
- La quasi totalità dei blocchi di elaborazione prevede di **assegnare** il risultato di un'operazione (o un dato) a una **variabile**.
- Con **variabile** intendiamo una «scatola» che è in grado di contenere numeri o altri dati e a cui è possibile fare riferimento, ad esempio dandole un nome come abbiamo fatto con i parametri di *input* e *output*.
- **Assegnare** significa inserire dentro una «scatola» un dato, ottenibile anche come risultato di un calcolo. Il simbolo utilizzato per l'assegnamento è ←

Blocco di Elaborazione

- Il concetto di assegnamento in informatica è profondamente diverso da quello di uguaglianza matematica!
- **In matematica**, quando due cose sono uguali significa che sono la stessa cosa. Quindi quando scriviamo $x = 5$ in matematica stiamo dicendo che x è 5 e in questo contesto x è e sarà sempre 5.
- Non vi è quindi alcun concetto temporale: non ha senso chiedersi quanto valeva x *prima*.
- **In informatica** all'assegnamento corrisponde un'operazione che viene eseguita in uno specifico momento.
- Quindi, quando scriviamo $x \leftarrow 5$ stiamo dicendo che prima nella «scatola» x c'era qualcosa, ma dopo l'assegnamento c'è il valore 5.
- In seguito, il valore contenuto in x potrebbe cambiare ulteriormente. **Il concetto di tempo assume quindi un ruolo fondamentale.**

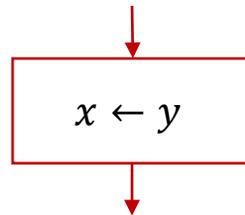
Blocco di Elaborazione

- Scrivere il nome di una variabile a sinistra dell'assegnamento (\leftarrow) significa che il valore di quello che si trova a destra verrà inserito nella «scatola» corrispondente alla variabile stessa.
- Ad esempio, il blocco elaborazione che segue mette il valore 5 dentro x :



Blocco di Elaborazione

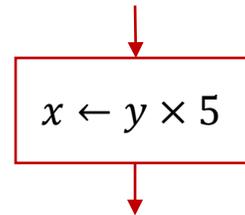
- Scrivere il nome di una variabile a destra dell'assegnamento (\leftarrow) significa guardare il valore contenuto nella variabile e utilizzarlo per fare dei calcoli.
- Il contenuto della variabile non viene modificato dall'operazione di lettura: «guardare» dentro una «scatola» non ne cambia il contenuto.
- Ad esempio, il blocco elaborazione che segue copia dentro x il valore contenuto in y :



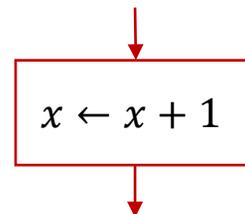
- y contiene ancora il suo valore precedente.

Blocco di Elaborazione

- Il blocco elaborazione che segue guarda il contenuto di y , lo moltiplica per 5 e mette il risultato dentro x :



- Il blocco elaborazione che segue legge il contenuto di x , gli somma 1, e mette il risultato nuovamente dentro x :



- Se x prima valeva 3, dopo l'operazione x varrà 4

Terminologia

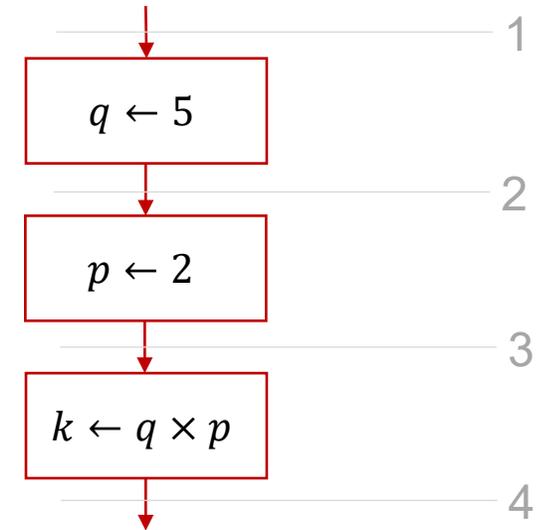
- L'operazione di «guardare» il contenuto di una «scatola» sarà identificata con il termine **lettura** del contenuto di una variabile, o più semplicemente lettura di una variabile.
- I termini **scrivere** o **assegnare** verranno utilizzati per riferirsi all'operazione di «mettere» qualcosa dentro una «scatola».
- Il contenuto di una «scatola» è il suo **valore**.
- Dopo aver eseguito $x \leftarrow 4$, si dice che x vale 4. Che cosa abbiamo fatto? Abbiamo assegnato 4 a x .
- Se eseguiamo $x \leftarrow x + 1$, stiamo assegnando a x il risultato di $x + 1$. Come lo si calcola? 1) Si legge x ; 2) Si incrementa il valore letto di 1; 3) Si scrive il risultato in x .

La Sequenza di Operazioni

- Uno dei concetti fondamentali in informatica è quello di sequenza, ovvero un elenco di operazioni da eseguire una dopo l'altra.
- Al termine dell'esecuzione di una operazione si procede con la successiva, e così via fino a quando le operazioni non sono terminate.
- Questa modalità di esecuzione delle operazioni è detta **sequenziale** in contrapposizione ad una esecuzione **parallela** in cui più attività vengono eseguite contemporaneamente.
- La sequenzialità è intrinseca nella rappresentazione mediante diagrammi di flusso.

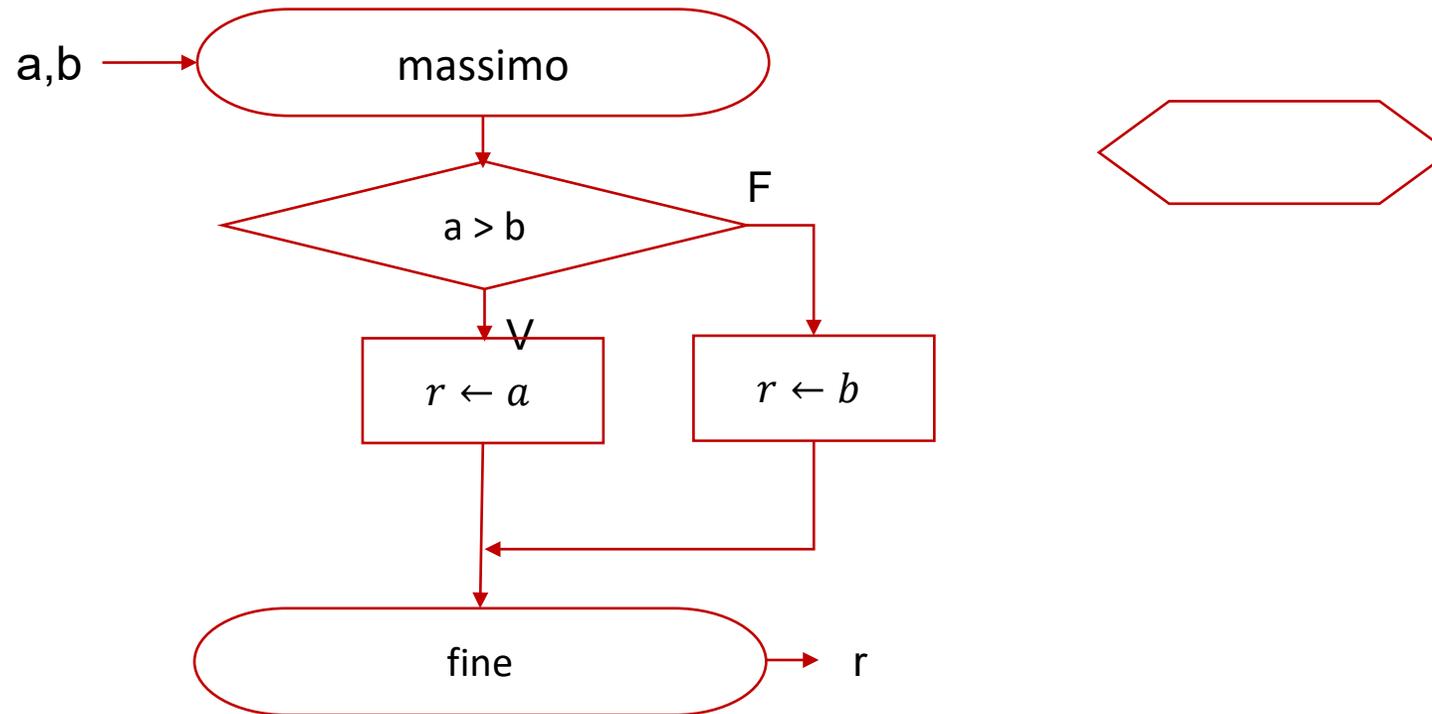
La Sequenza di Operazioni

- Un esempio di sequenza è riportato a destra.
- Al *tempo* 1 non possiamo dire cosa vale q non possiamo dire cosa vale p , e non possiamo dire cosa vale k .
- Al *tempo* 2 possiamo dire cosa vale q , ovvero 5, ma non possiamo dire cosa valgono p o k .
- Al *tempo* 3 possiamo dire cosa valgono q (5) e p (2), ma non possiamo dire cosa vale k .
- Al *tempo* 4 possiamo dire che q vale 5, p vale 2 e che k vale 10, risultato dell'operazione $q \times p$ (5×2).

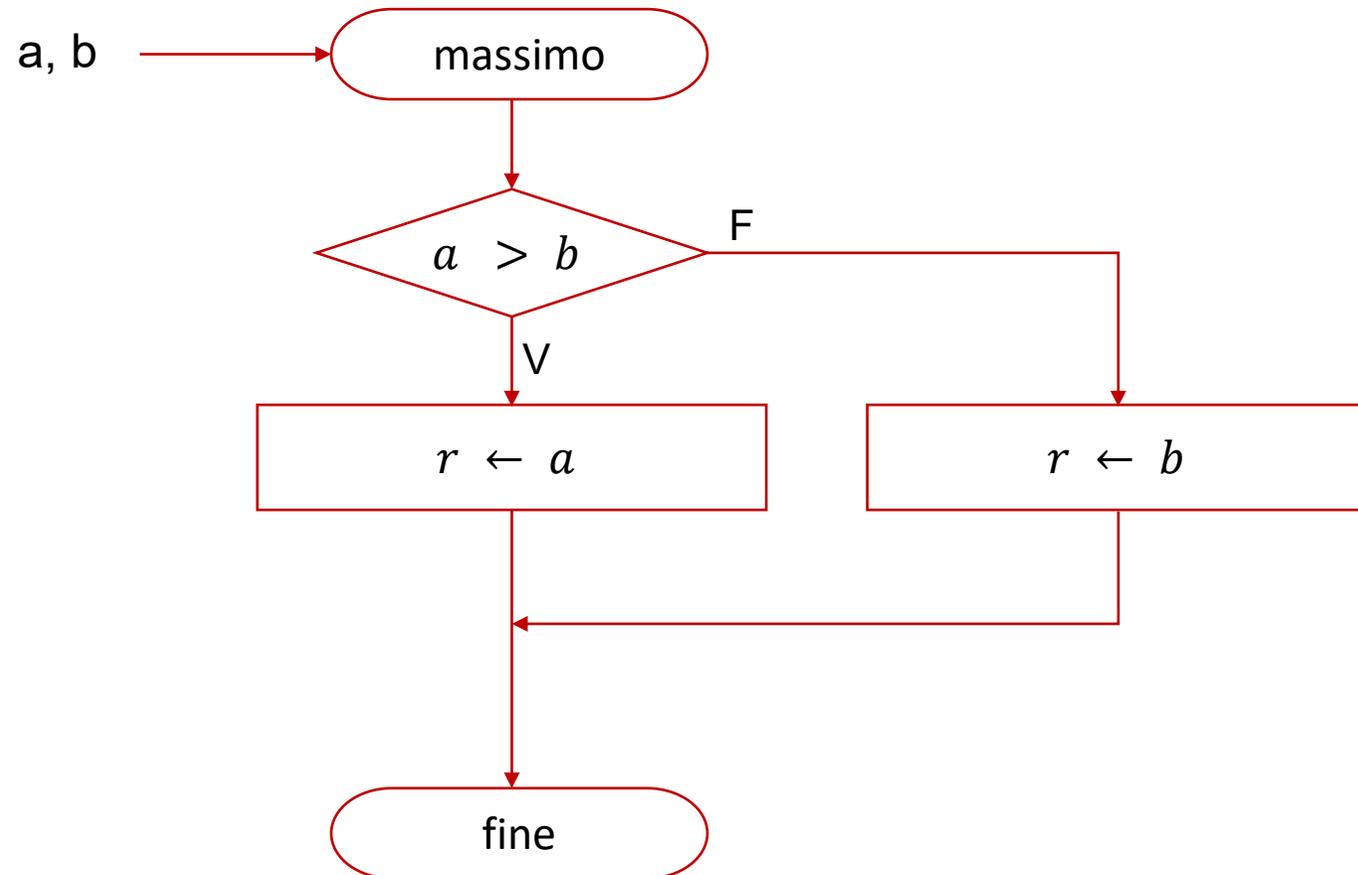


Esempio 1

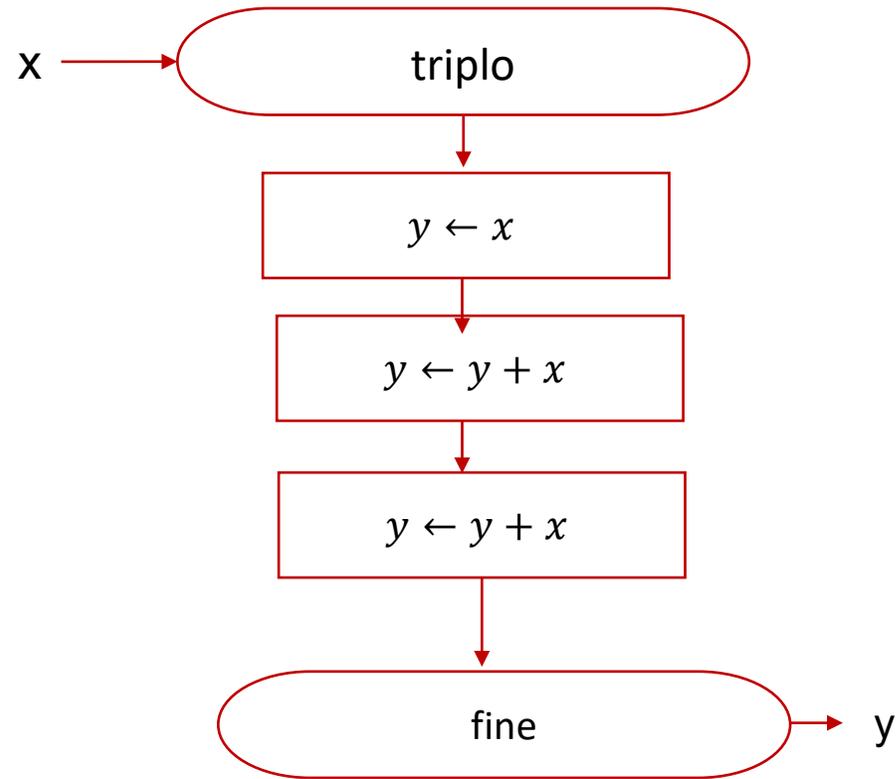
- Dati due numeri a e b scrivere il diagramma di flusso dell'algoritmo che trova il massimo e lo salva in r .
- Ad esempio, se i numeri a e b fossero rispettivamente 4 e 7 il programma dovrebbe salvare in r il valore di b , ovvero 7.



Esempio 1



- Disegnare il diagramma di flusso del programma *triplo* che dato un numero x ritorna $y = 3 \cdot x$



$[var] \leftarrow [numero]$

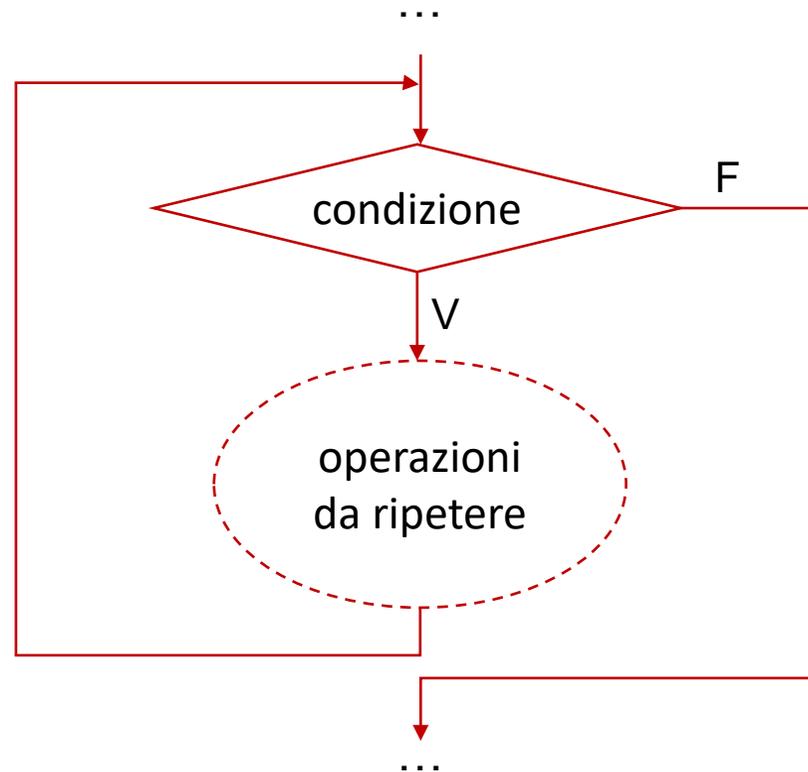
$[var] \leftarrow [var1] + [var2]$

$[var] \leftarrow [var]$

Come Modellare la Ripetizione

- A volte è necessario **ripetere** un insieme di operazioni più volte.
- Purtroppo non esiste un blocco specifico nei diagrammi di flusso per modellare questo concetto.
- Ad ogni modo è possibile comporre i blocchi a disposizione per ottenere il risultato desiderato, ovvero la ripetizione di una (o più) istruzioni un numero fissato di volte.

Come Modellare la Ripetizione

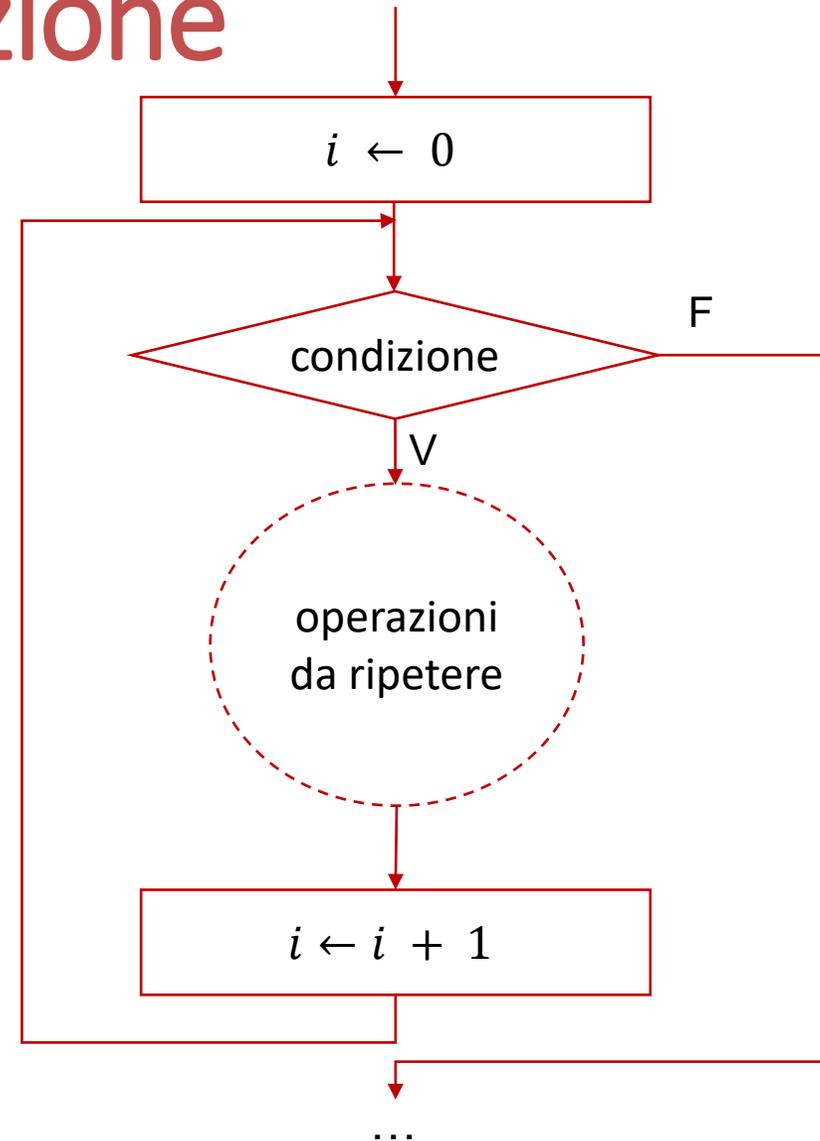


Come Modellare la Ripetizione

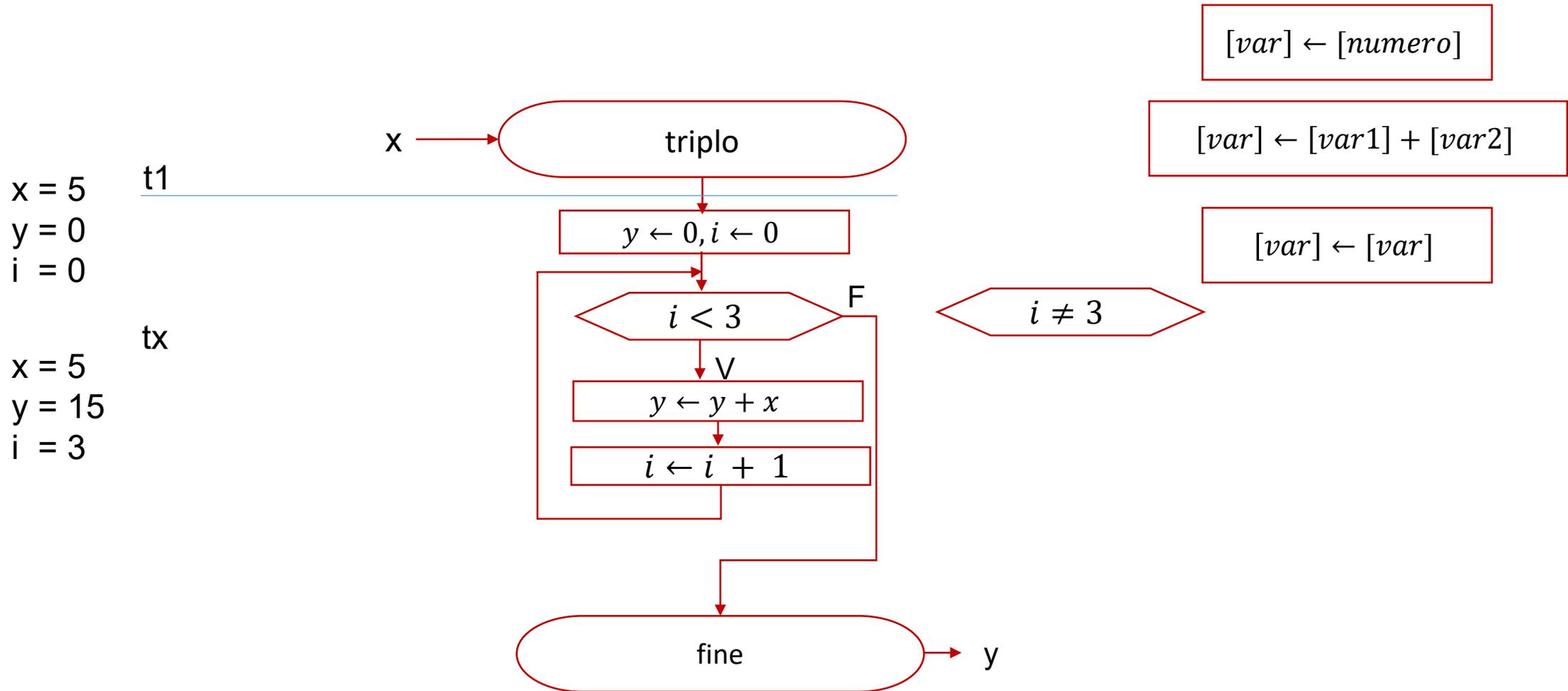
- Il concetto di ripetizione richiede di contare il numero di volte in cui le istruzioni sono state ripetute, in modo tale da potersi fermare (interrompere la ripetizione) quando necessario.
- Contare significa:
 - definire uno stato iniziale, ad esempio $i \leftarrow 0$ (questa fase è chiamata in informatica fase di **inizializzazione**)
 - cambiare lo stato dopo aver ripetuto le istruzioni, ad esempio $i \leftarrow i + 1$ (questa fase è chiamata in informatica fase di **aggiornamento**);

Come Modellare la Ripetizione

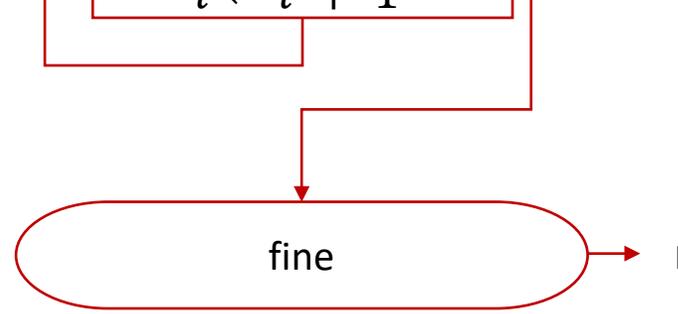
- Tipicamente la condizione di ripetizione dovrà essere correlata alla variabile utilizzata per contare, i nell'esempio a destra.
- Se ad esempio dovessi ripetere le operazioni n volte la condizione dovrebbe essere $i < n$.
- La **condizione**, insieme all'**inizializzazione** e all'**aggiornamento**, rappresenta uno degli elementi chiave della ripetizione.



- Disegnare il diagramma di flusso del programma *triplo* che dato un numero x ritorna $y = 3 \cdot x$

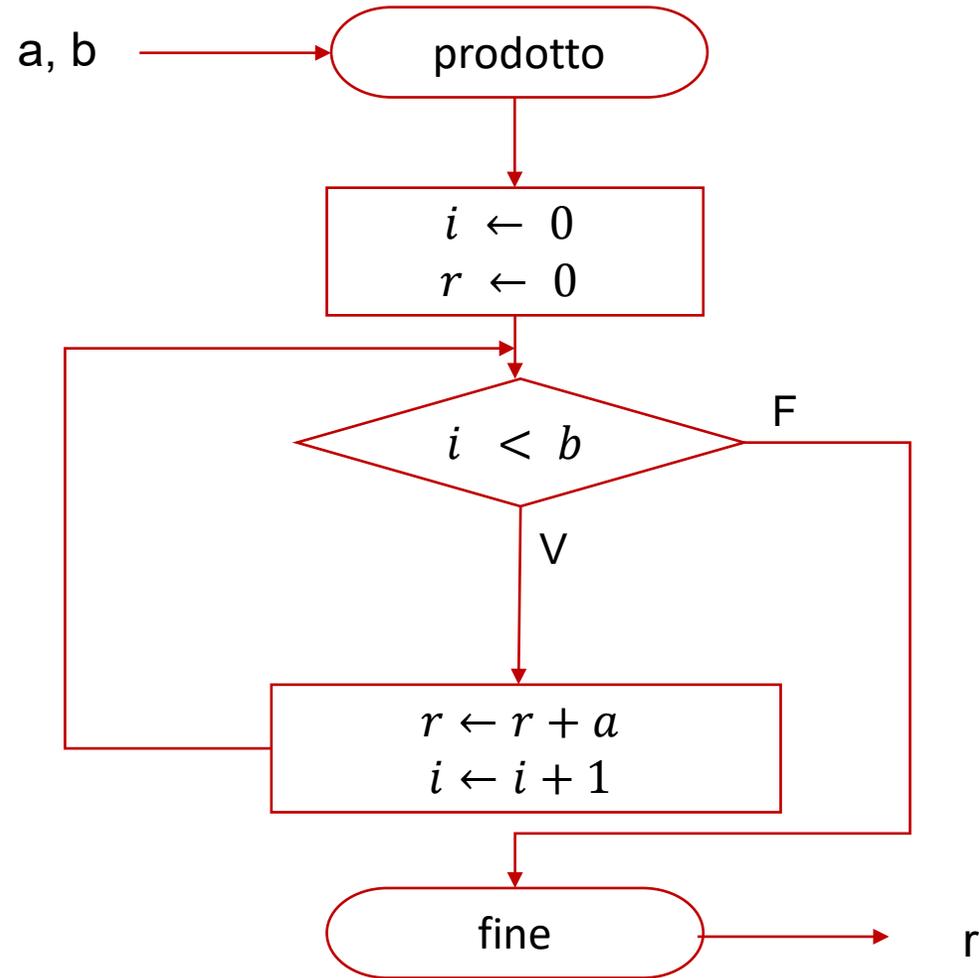


Esempio 2



- Dati due numeri a e b scrivere il diagramma di flusso dell'algoritmo che calcola il prodotto dei numeri ($a \times b$) utilizzando solo la somma. L'algoritmo deve salvare il risultato in r . Si assuma che b sia sempre un numero intero positivo, al più nullo. Il numero a può essere un qualunque numero reale.
- Ad esempio, se i numeri a e b fossero rispettivamente 4.1 e 3 l'algoritmo dovrebbe calcolare $r = 4.1 + 4.1 + 4.1$ ovvero sommare 4.1 (a) per 3 (b) volte.

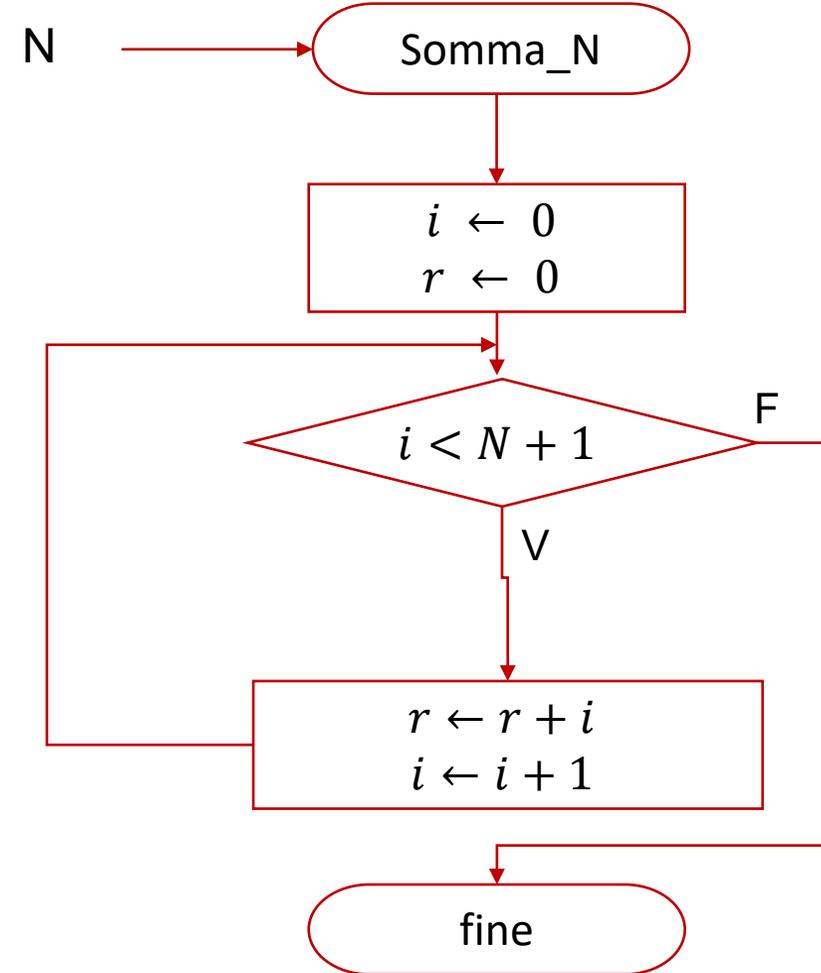
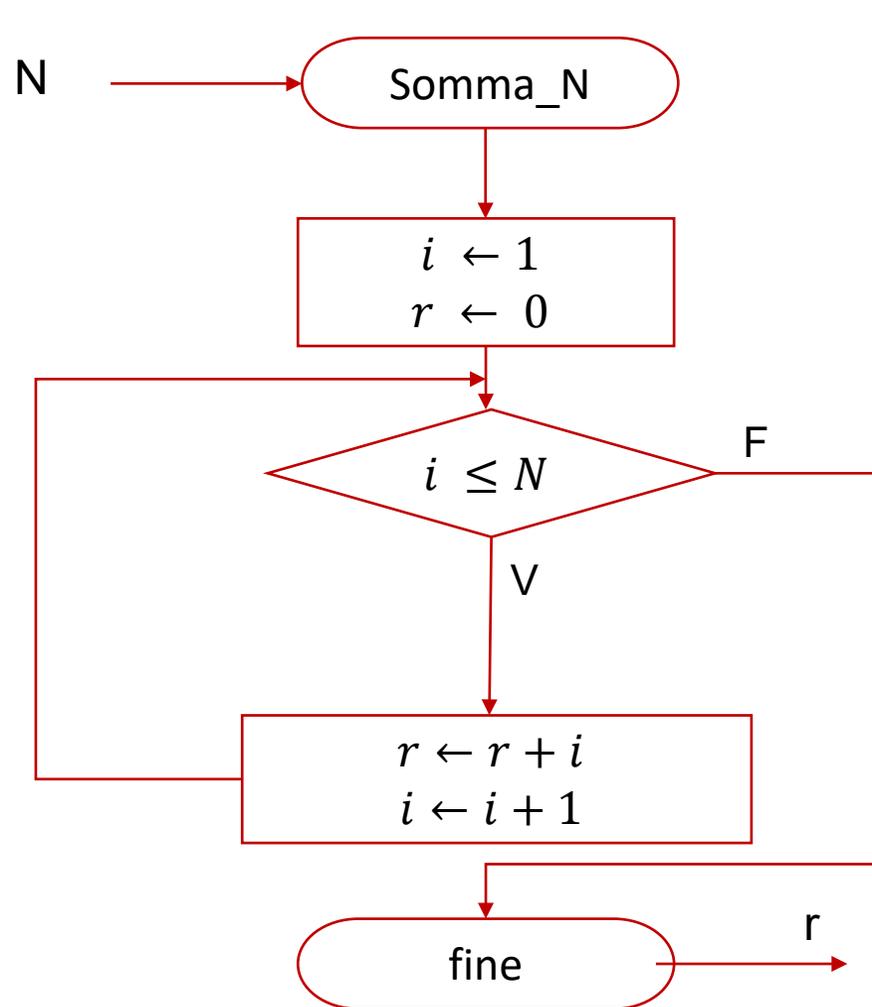
Esempio 2



Somma Naturali

- Scrivere il diagramma di flusso di un programma che dato N calcola $r = \sum_{i=1}^N i$, ovvero la somma dei primi N numeri naturali.
- Esempio: se $N = 5$, $r = 1 + 2 + 3 + 4 + 5 = 15$
- $r = 0 + 1 + 2 + 3 + 4 + 5 = 15$
- $r = 1 + 2 + 3 + 4 + 5 = 15$

Somma Naturali

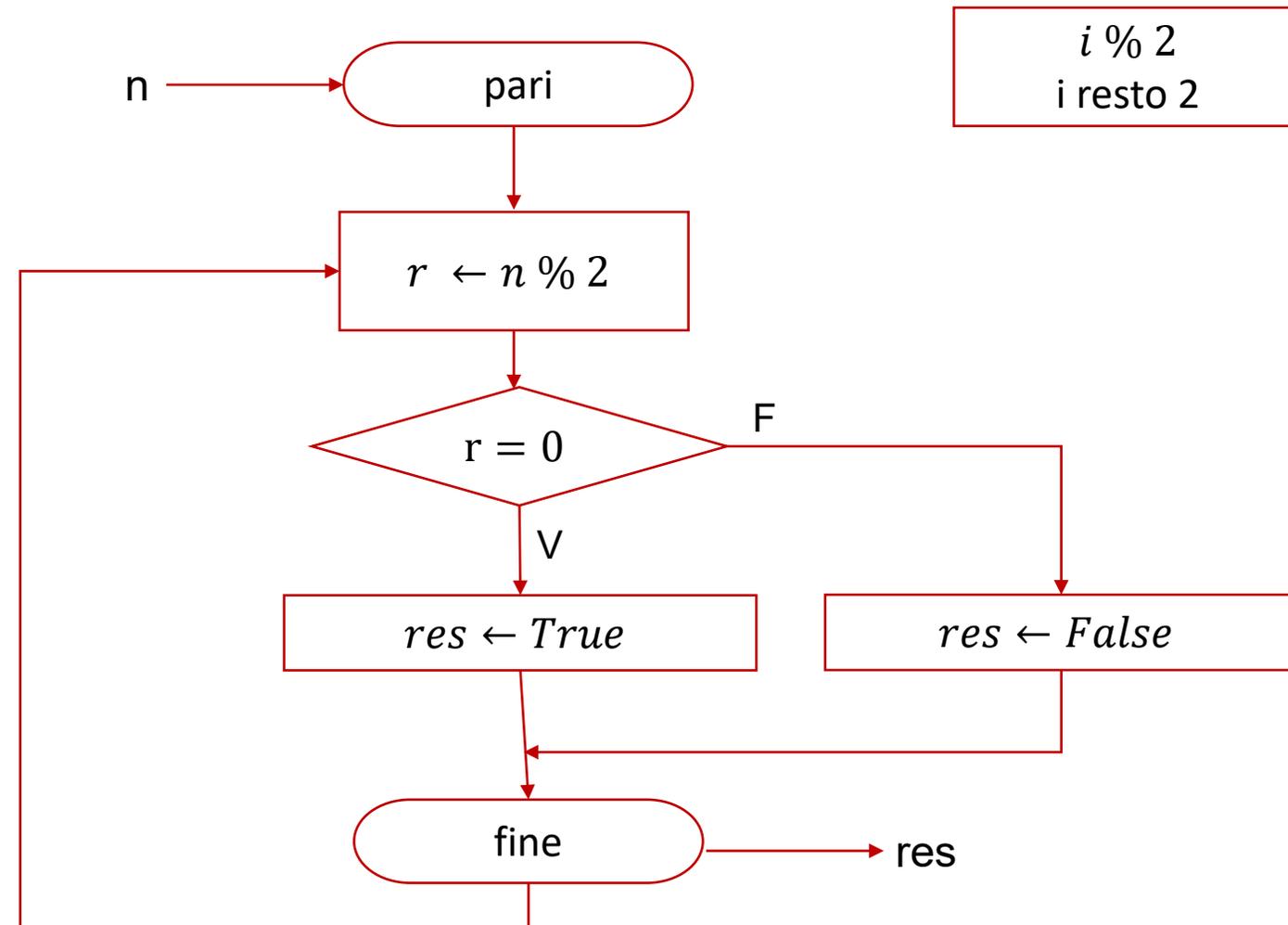


Formula di Gauss

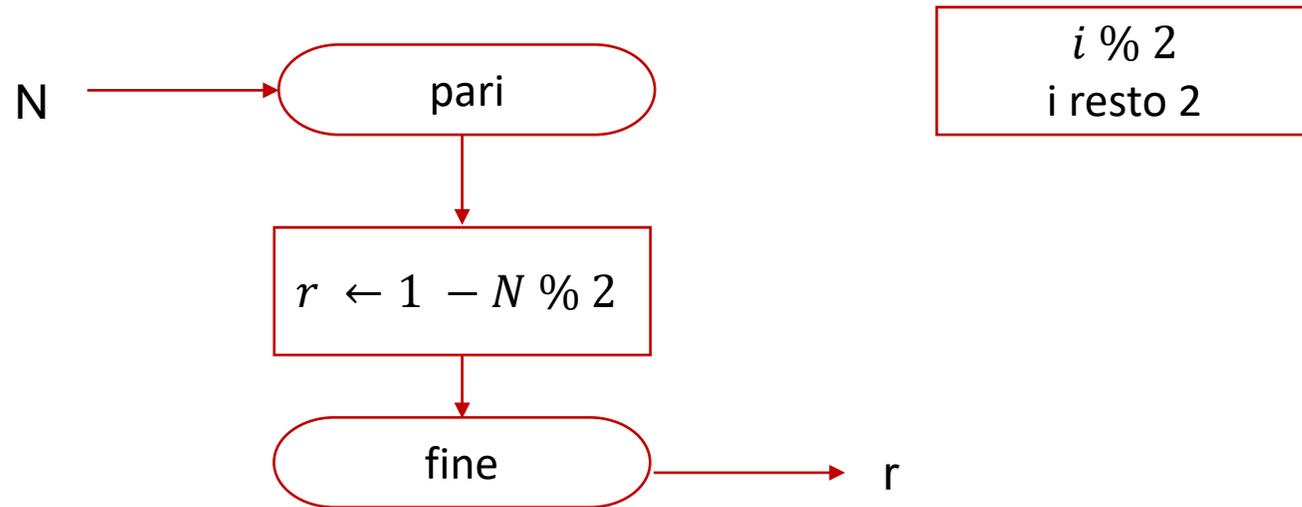
$$r = \sum_{i=1}^N i = \frac{N * (N + 1)}{2}$$

- Dato un numero intero N ritornare $r = True$ se N è pari, $r = False$ se N è dispari.

- $7\%4 = 3$ $7/4 = 1$
- $N = q * D + r$



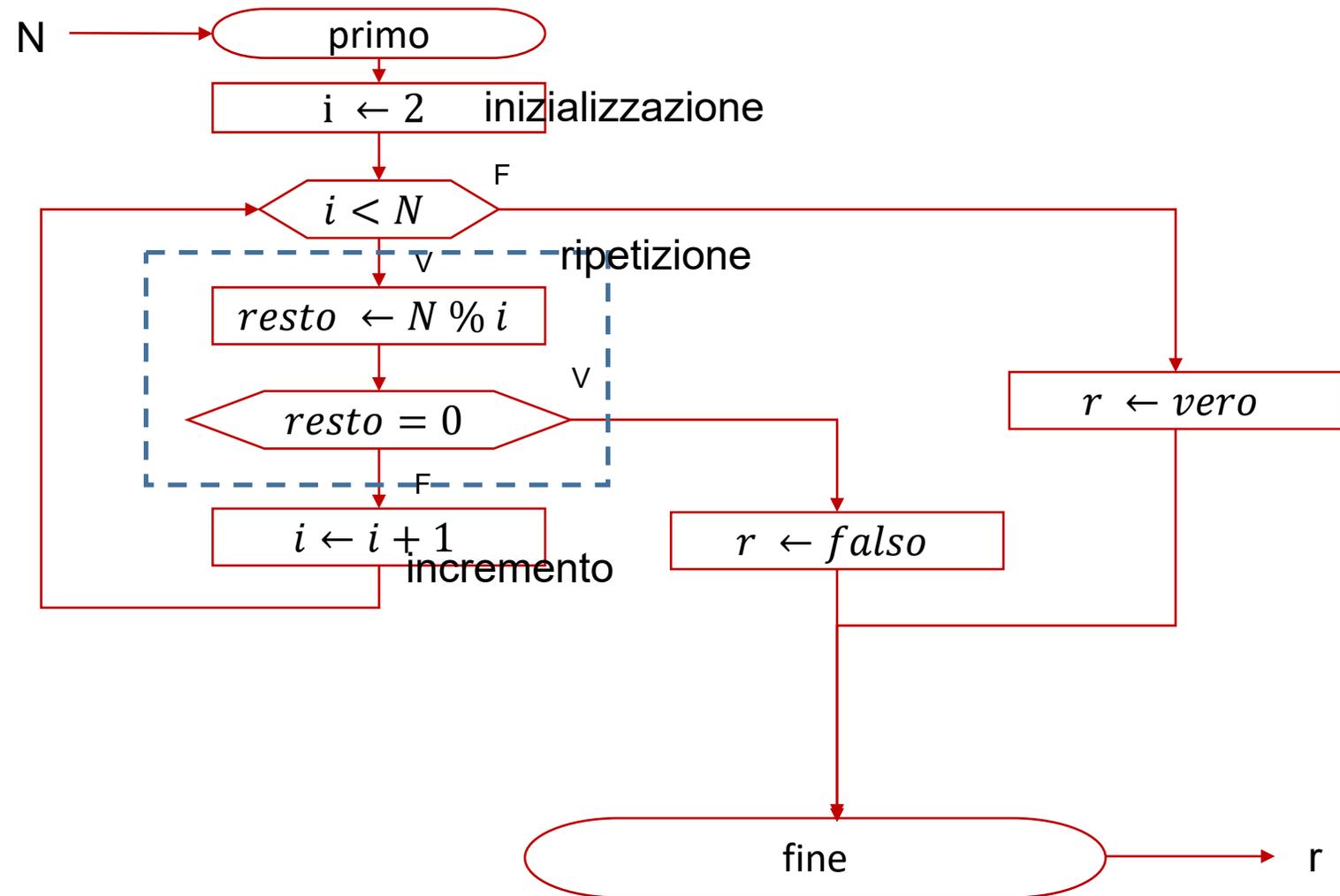
- Dato un numero intero N ritornare $r = 1$ se N è pari, $r = 0$ se N è dispari.



- $7 \% 4 = 3$ $7 / 4 = 1$
- $N = q * D + r$

- Dato un numero N ritorna $r = 1$ se N è primo, $r = 0$ se N non è primo.
- $\% = \text{resto} = \text{mod}$
- 1728354563
- $1728354563 \% 2 \neq 0$
- $1728354563 \% 3 \neq 0$
- $1728354563 \% 4 \neq 0$
- ...
- $1728354563 \% 1728354562 \neq 0$
- $N \% N \neq 0$

- Dato un numero N ritorna $r = \text{vero}$ se N è primo, $r = \text{falso}$ se N non è primo.

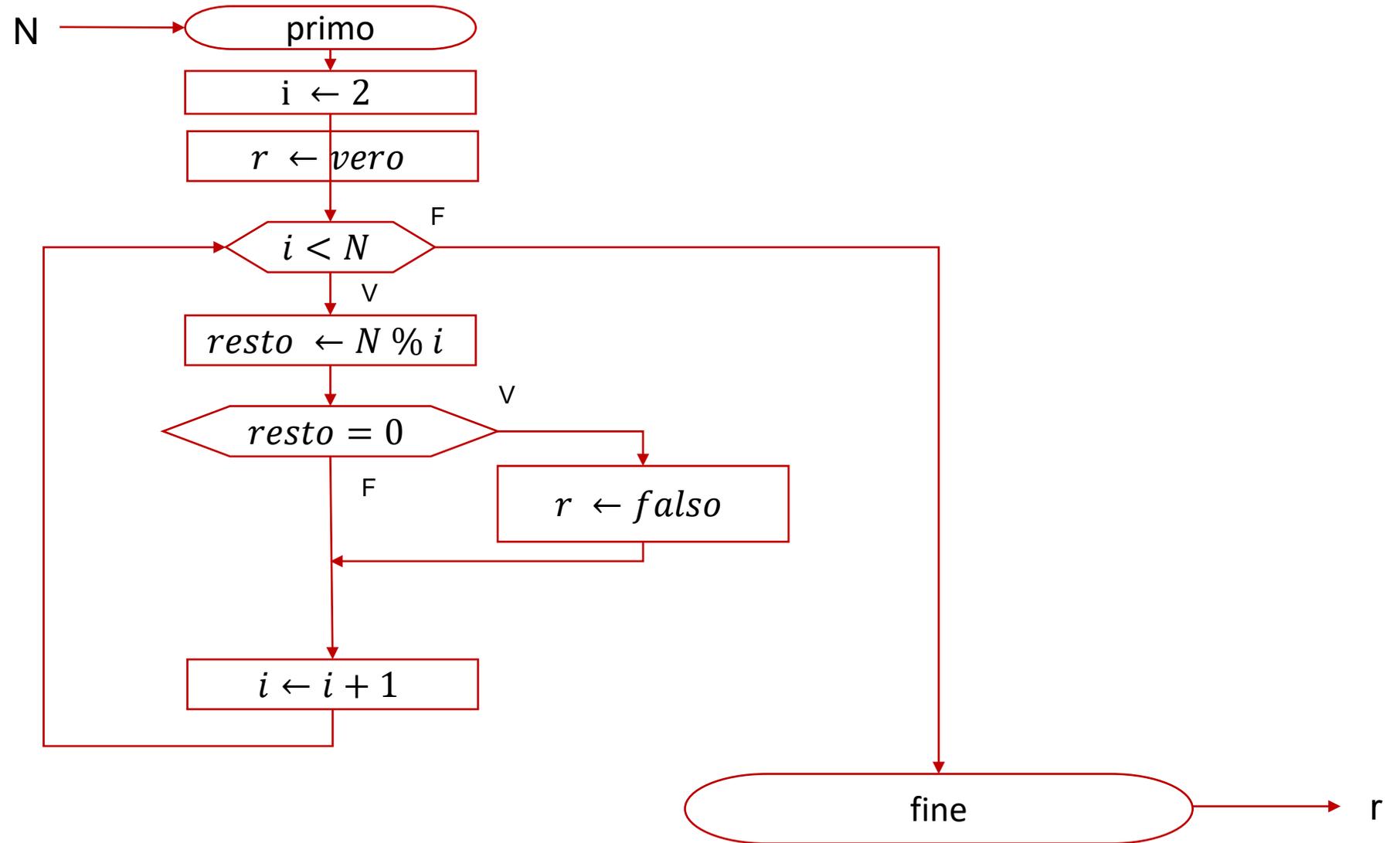


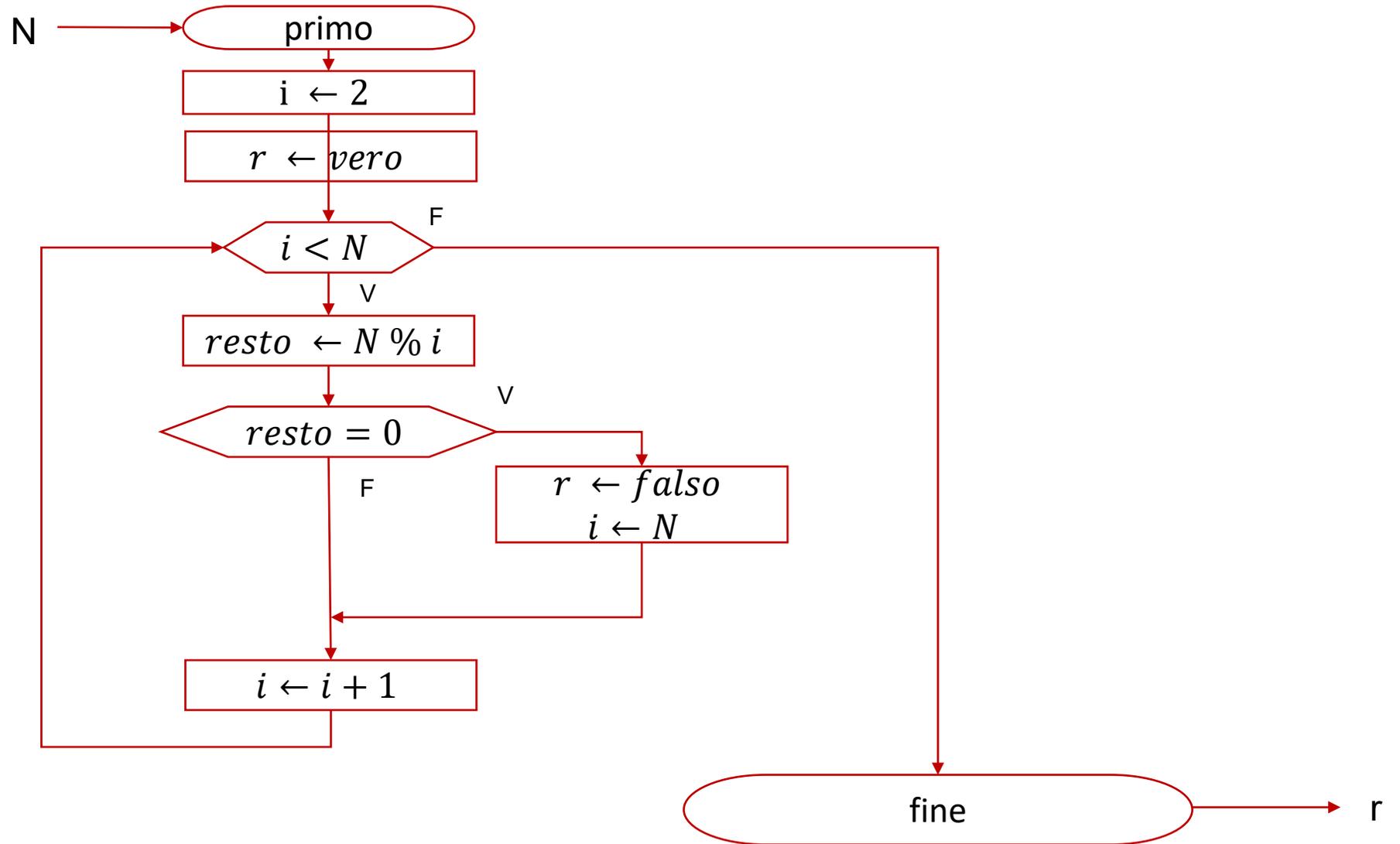
N = 1728467238

i = 3

r = falso

resto = 0





N = 1728467238

i = 3

r = falso

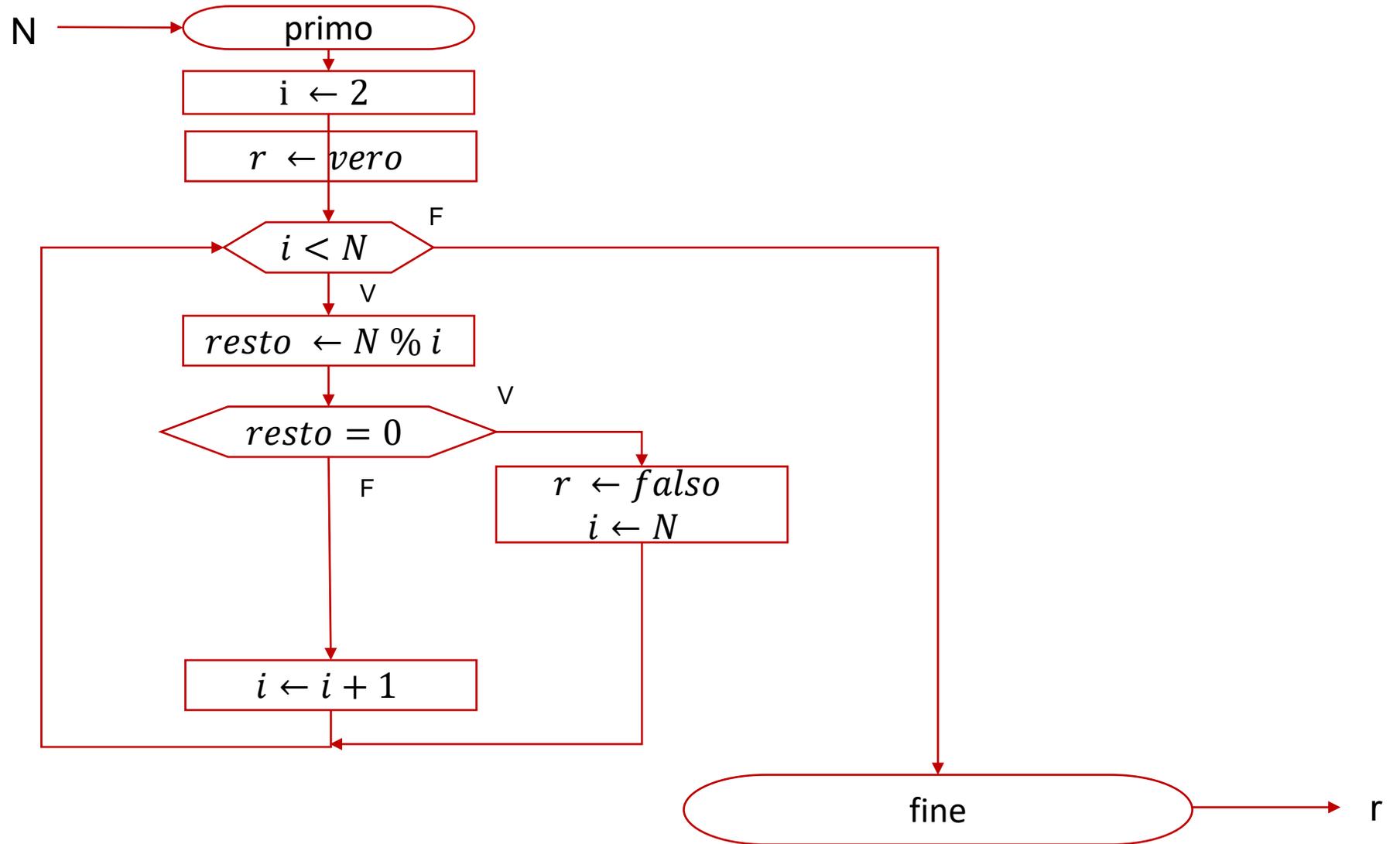
resto = 0

N = 1728467238

i = 3

r = falso

resto = 0



N = 1728467238
i = 3
r = false
resto = 0

