# Informatica

CdL in Matematica

Parte 5

Roberto Zunino

# Correttezza dei programmi

# Correttezza

Uno dei problemi più importanti nella programmazione è quello di dimostrare che la *semantica* di un dato programma corrisponda effettivamente a quella voluta, cioè alla *specifica*.

Come esempio, prendiamo un problema molto semplice: lo scambio del valore di due variabili. Abbiamo visto che lo possiamo fare col comando

$$c = \qquad t := x;\ x := y;\ y := t$$

ma come dimostrarlo?

Usare la definizione della semantica (per es. big step) è teoricamente possibile, ma non molto pratico.

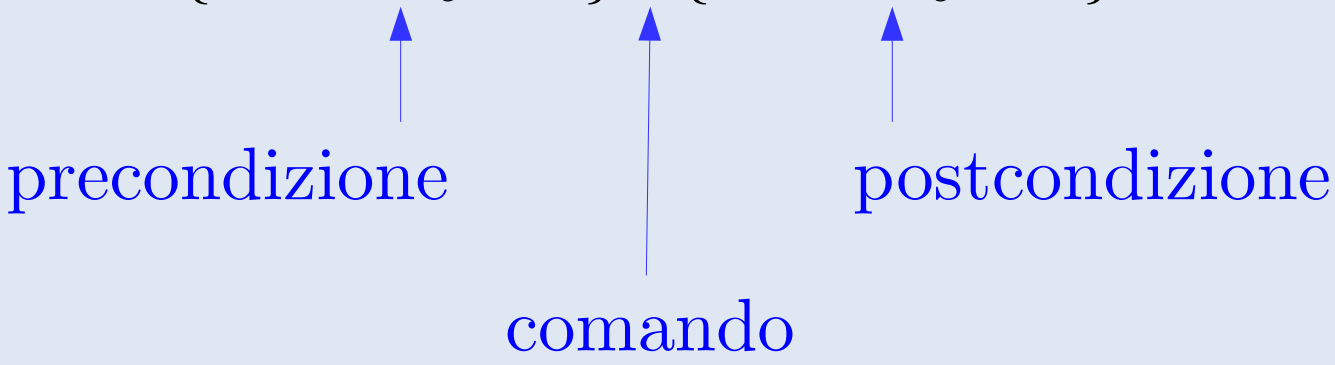
# Triple di Hoare

Proviamo intanto a scrivere meglio la specifica:

> Se si parte da uno stato in cui $x$ vale $a$ e $y$ vale $b$ allora dopo avere eseguito $c$ e raggiunto uno stato finale, in tale stato $x$ vale $b$ e $y$ vale $a$.

In simboli, la specifica di sopra viene scritta usando le cosiddette triple di Hoare:

$$\{x = a \wedge y = b\}\ c\ \{x = b \wedge y = a\}$$



# Variabili di Specifica

La *tripla di Hoare* che abbiamo visto prima

$$\{x = a \wedge y = b\}\ c\ \{x = b \wedge y = a\}$$

ha una piccola ambiguità, in quanto potrebbe essere fraintesa come:

> Se si parte da uno stato in cui le variabili $x$ e $a$ sono uguali, e in cui le variabili $y$ e $b$ sono uguali
> ...

L'ambiguità nasce dal fatto che nella precondizione $x = a$ stiamo mischiando variabili "di programma" ($x$) e variabili "matematiche" ($a$) che non si riferiscono ai valori usati nel programma ma servono sono ad indicare valori arbitrari.

# Variabili di Specifica

Queste "variabili non di programma" vengono chiamate variabili di specifica o anche metavariabili.

Per evitare confusione, si adotta la convenzione di scrivere le variabili di programma con lettere *minuscole*, e le variabili di specifica con lettere *maiuscole* nelle triple di Hoare.

Per esempio:

$$\{x = A \wedge y = B\}\ c\ \{x = B \wedge y = A\}$$

# Proprietà sugli Stati

**Def.** Sia $P$ una proprietà sugli stati. Scriviamo $\sigma \models P$ per indicare che lo stato $\sigma$ *soddisfa* la prioprietà $P$.

**Nota.** Per definire formalmente la relazione $\models$ dovremmo definire precisamente il "linguaggio" formale in cui vengono scritte le proprietà $P$. Non lo faremo, e scriveremo tali $P$ usando la notazione matematica usuale, affidandoci all'intuizione quando dovremo capire se $\sigma$ soddisfa $P$ o meno.

# Validità di una Tripla

**Def.** Una tripla di Hoare

$$\{P\}\, c\, \{Q\}$$

viene detta *valida* se e solo se

$$\sigma \vDash P \ \wedge\ \langle c\ ,\ \sigma\rangle \to_b \sigma' \implies \sigma' \vDash Q$$

per ogni $\sigma, \sigma'$.

In termini informali, è valida quando, se si parte una uno stato $\sigma$ che soddisfa la precondizione e si termina in $\sigma'$, allora $\sigma'$ soddisfa la postcondizione.

La validità di una tripla viene denotata come segue:

$$\vDash\ \{P\}\, c\, \{Q\}$$

# Correttezza (di nuovo)

Abbiamo dato una definizione di specifica formale usando le triple di Hoare.

Abbiamo capito che vogliamo poter dimostrare la validità di una tripla data.

Non abbiamo però ancora imparato come farlo in pratica.

# Regole di Inferenza per le Triple di Hoare

# Sostituzione

**Def.** Data una proprietà formale $P$, indichiamo con $P\{e/x\}$ il rimpiazzamento simbolico della variabile $x$ dentro $P$ con l'espressione $e$.

Esempi.

$$
\begin{aligned}
(x = 5)\{4/x\} &= 4 = 5 \\
(x + 1 \leq y)\{x + 2/x\} &= (x + 2) + 1 \leq y \\
(\textstyle\sum_{0<J<10} x^J = y)\{A + 2/x\} &= (\textstyle\sum_{0<J<10} (A + 2)^J = y)
\end{aligned}
$$

# Regole per le Triple

**Def.** Definiamo ricorsivamente la relazione $\{P\}\ c\ \{Q\}$ tra due propretà e un comando.

$$\frac{}{\{P\}\ \mathsf{skip}\ \{P\}}\,[Skip]$$

# Regole per le Triple

$$\frac{}{\{P\{e/x\}\}\ x\ :=\ e\ \{P\}}\ [Let]$$

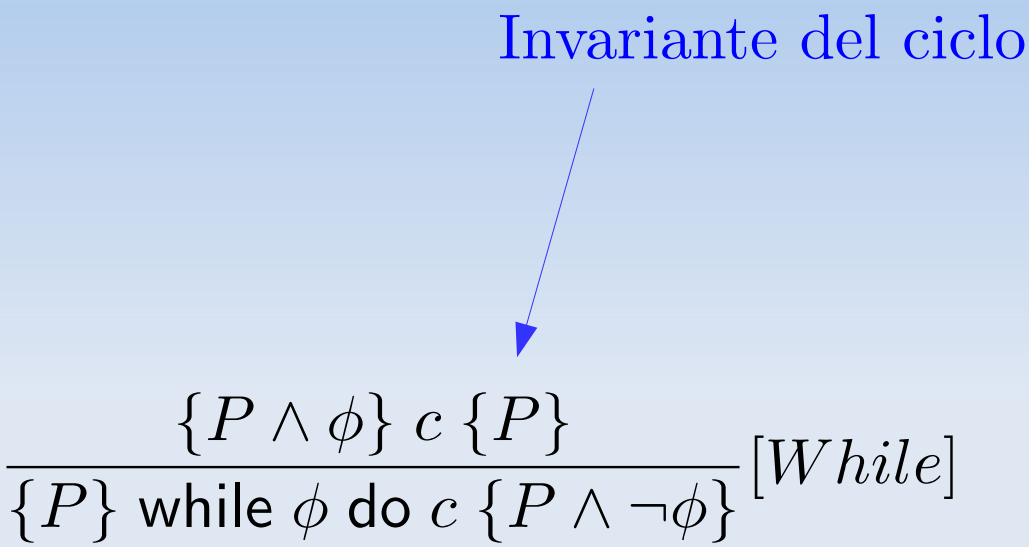
# Regole per le Triple

$$\frac{\{P\}\ c_1\ \{Q\} \qquad \{Q\}\ c_2\ \{R\}}{\{P\}\ c_1; c_2\ \{R\}}[Comp]$$

# Regole per le Triple

$$\frac{\{P \wedge \phi\}\ c_1\ \{Q\} \qquad \{P \wedge \neg\phi\}\ c_2\ \{Q\}}{\{P\}\ \text{if}\ \phi\ \text{then}\ c_1\ \text{else}\ c_2\ \{Q\}}\,[If]$$

# Regole per le Triple



# Regole per le Triple

$$\frac{P \implies P' \qquad \{P'\}\ c\ \{Q'\} \qquad Q' \implies Q}{\{P\}\ c\ \{Q\}}\,[PrePost]$$

(Fine delle regole)

# Notazione

Quando una tripla è derivabile dalle regole di inferenza, scriviamo

$$\vdash \{P\}\ c\ \{Q\}$$

in modo da distinguere la relazione definita ricorsivamente da quella di validità vista prima (che si scrive $\models \{P\}\ c\ \{Q\}$).

# Teorema di Correttezza

**Teorema. (correttezza)**

$$\vdash \{P\}\ c\ \{Q\} \qquad \Longrightarrow \qquad \vDash \{P\}\ c\ \{Q\}$$

In altre parole, le regole viste prima sono corrette: se esiste una derivazione per una tripla, essa è valida.

# Teorema di Correttezza

Rimandiamo la dimostrazione.

Prima, vediamo qualche esempio.

# Esempio: Let+Comp

Dimostriamo la correttezza dello scambio di variabili:

$$\{x = A \wedge y = B\}\ t := x;\ x := y;\ y := t\ \{y = A \wedge x = B\}$$

Usando le regole costruisco una derivazione:

$$\dfrac{\dfrac{D}{\{x = A \wedge y = B\}\ t := x;\ x := y\ \{t = A \wedge x = B\}}\quad \dfrac{}{\{t = A \wedge x = B\}\ y := t\ \{y = A \wedge x = B\}}[Let]}{\{x = A \wedge y = B\}\ t := x;\ x := y;\ y := t\ \{y = A \wedge x = B\}}[Comp]$$

dove:

$$D = \dfrac{\dfrac{}{\{x = A \wedge y = B\}\ t := x\ \{t = A \wedge y = B\}}[Let]\quad \dfrac{}{\{t = A \wedge y = B\}\ x := y\ \{t = A \wedge x = B\}}[Let]}{\{x = A \wedge y = B\}\ t := x;\ x := y\ \{t = A \wedge x = B\}}[Comp]$$

# Notazione Compatta: Comp

In futuro per le catene di composizioni

$$c_1; c_2; c_3; \ldots$$

scriveremo le triple associate in una notazione più compatta:

$$\{P_0\}c_1; \{P_1\}c_2; \{P_2\}c_3; \{P_3\} \ldots$$

Questa notazione indica una ripetuta applicazione della regola $[Comp]$ come visto nella derivazione dell'esempio.

# Esempio

La precedente dimostrazione di validità, nella notazione compatta:

$$
\begin{array}{l}
\{x = A \wedge y = B\} \\
t := x; \\
\{t = A \wedge y = B\} \\
x := y; \\
\{t = A \wedge x = B\} \\
y := t \\
\{y = A \wedge x = B\}
\end{array}
$$

Si noti come, data l'asserzione in ultima riga, è possibile ricavare "a ritroso" le asserzioni intermedie tramite sostituzione.

# Notazione Compatta: PrePost

Data una tripla

$$\{P\}$$
$$c$$
$$\{Q\}$$

per rafforzare la precondizione o indebolire la postcondizione tramite la PrePost, scriviamo, rispettivamente:

$$\{P'\}$$
$$\{P\}$$
$$c$$
$$\{Q\}$$
$$\{Q'\}$$

e dimostriamo separatamente $P' \implies P$ e $Q \implies Q'$.

# Notazione Compatta: PrePost

Quando $Q' = Q$ o $P' = P$ abbreviamo come segue:

| | |
|---|---|
| $\{P'\}$ | $\{P\}$ |
| $\{P\}$ | $c$ |
| $c$ | $\{Q\}$ |
| $\{Q\}$ | $\{Q'\}$ |

In pratica, nella notazione compatta, tutte le volte che compaiono due asserzioni $\{R\}\{S\}$ in due righe di fila, stiamo usando una PrePost, e per farlo dobbiamo dimostrare a parte $R \implies S$.

# Esercizio

Dimostriamo che

$$
\begin{aligned}
&\{a = A \wedge b = B\} \\
&a := a + b; \\
&b := a - b; \\
&a := a - b \\
&\{a = B \wedge b = A\}
\end{aligned}
$$

Per farlo, basta partire dal fondo e sostituire all'indietro, in modo meccanico.

# Soluzione

$\{a = A \wedge b = B\} \qquad (1)$
$\{(a + b) - ((a + b) - b) = B \wedge (a + b) - b = A\}$
$a := a + b;$
$\{a - (a - b) = B \wedge a - b = A\}$
$b := a - b;$
$\{a - b = B \wedge b = A\}$
$a := a - b$
$\{a = B \wedge b = A\}$

Si noti come al punto (1) abbiamo bisogno di una PrePost. L'implicazione è immediata da dimostrare.

# Esempio: If+PrePost

Dimostriamo che

$$\{n \text{ pari}\} \text{ if } n > 3 \text{ then } n := n + 7 \text{ else } n := 5 \{n \text{ dispari } \geq 5\}$$

Procediamo così:

$$\frac{D1 \qquad D2}{\{n \text{ pari}\} \text{ if } n > 3 \text{ then } n := n + 7 \text{ else } n := 5 \{n \text{ dispari } \geq 5\}}[If]$$

$$D1 = \frac{\dfrac{}{\{(n+7) \text{ dispari} \geq 5\}\ n := n + 7\ \{n \text{ dispari } \geq 5\}}[Let]}{\{n \text{ pari} \wedge n > 3\}\ n := n + 7\ \{n \text{ dispari } \geq 5\}}[PrePost]$$

$$D2 = \frac{\dfrac{}{\{5 \text{ dispari } \geq 5\}\ n := 5\ \{n \text{ dispari } \geq 5\}}[Let]}{\{n \text{ pari} \wedge n \leq 3\}\ n := 5\ \{n \text{ dispari } \geq 5\}}[PrePost]$$

implicazioni per la PrePost

# Notazione Compatta

Useremo anche per l'if una notazione più compatta:

$\{n \text{ pari}\}$
if $n > 3$ then
    $\{n \text{ pari} \wedge n > 3\}$
    $\{(n+7) \text{ dispari} \wedge (n+7) \geq 5\}$
    $n := n + 7$
else
    $\{n \text{ pari} \wedge n \leq 3\}$
    $\{5 \text{ dispari} \wedge 5 \geq 5\}$
    $n := 5$
$\{n \text{ dispari} \wedge n \geq 5\}$

Dalla notazione di sopra è semplice ricavare una derivazione con la regola $[If]$.

# Esercizio

Si dimostri la validità della seguente tripla:

$$
\begin{array}{l}
\{n \geq 0 \wedge n \text{ pari}\} \\
\textsf{if } n \leq 0 \textsf{ then} \\
\qquad n := n + 20 \\
\textsf{else} \\
\qquad n := n + 9 \\
\{n \geq 11\}
\end{array}
$$

# Esercizio

Si dimostri la validità della seguente tripla:

$\{n \geq 5\}$
if $n = 3$ then
    $n := 0$
else
    $n := n + 9$
$\{n \geq 14\}$

# Esempio: While

Calcoliamo $5 \cdot n$ tramite somma ripetuta:

$$
\begin{array}{l}
\{n = N\} \\
m := 0; \\
p := 0; \\
\textsf{while } m \neq n \textsf{ do} \\
\qquad p := p + 5; \\
\qquad m := m + 1 \\
\{p = 5 \cdot N\}
\end{array}
$$

Lo scriviamo direttamente nella notazione compatta, usando implicitamente la regola $While$ per le triple di Hoare.

# Soluzione

$$\frac{\{P \wedge \phi\}\ c\ \{P\}}{\{P\}\ \text{while}\ \phi\ \text{do}\ c\ \{P \wedge \neg\phi\}}$$

$\{n = N\}$
$\{0 = 5 \cdot 0 \wedge n = N\}$
$m := 0;$
$\{0 = 5 \cdot m \wedge n = N\}$
$p := 0;$
$\{INV : \ p = 5 \cdot m \wedge n = N\}$ *invariante*
while $m \neq n$ do
$\{INV \wedge m \neq n\}$
$\{p + 5 = 5 \cdot (m + 1) \wedge n = N\}$
$p := p + 5;$
$\{p = 5 \cdot (m + 1) \wedge n = N\}$
$m := m + 1$
$\{INV\}$ (d'ora in poi omettiamo questa riga)
$\{INV \wedge \neg(m \neq n)\}$
$\{q = 5 \cdot N\}$

# Sulle Invarianti

Trovare un'invariante che permetta di dimostrare la validità di un while è un problema difficile, in generale.

Non esiste un procedimanto meccanico per potere trovare una tale invariante, così come non esiste un procedimento meccanico per dimostrare enunciati. Ci vuole l'intuizione giusta, che va costruita facendo esercizi.

# Sulle Invarianti

Quando si è alla ricerca di un'invariante $INV$ per while $\phi$ do $c$ è bene pensare che:

- $INV$ deve valere subito prima di entrare nel while
- $INV$ deve valere subito dopo ogni esecuzione di $c$
- $INV$ deve valere subito dopo essere usciti dal while

Deve inoltre contenere "abbastanza informazione" per potere dimostrare la postcondizione desiderata ($q = 5 \cdot N$, nell'esempio precedente).
Scegliendo $INV =$ vero gestiamo facilmente il while, ma poi dobbiamo ricavare la postcondizione da $INV \wedge \neg\phi$, ovvero da solo $\neg\phi$, che di solito non basta.

# Esempio: Quadrato

Calcoliamo il quadrato di $n$ sommandolo $n$ volte:

$$
\begin{array}{l}
\{n = N\} \\
m := n; \\
q := 0; \\
\textsf{while } m \neq 0 \textsf{ do} \\
\qquad q := q + n; \\
\qquad m := m - 1 \\
\{q = N^2\}
\end{array}
$$

# Soluzione

$\{n = N\}$
$\{0 = n \cdot (n - n) \wedge n = N\}$
$m := n;$
$\{0 = n \cdot (n - m) \wedge n = N\}$
$q := 0;$
$\{INV : \ q = n \cdot (n - m) \wedge n = N\}$ *invariante*
while $m \neq 0$ do
    $\{INV \wedge m \neq 0\}$
    $\{q + n = n \cdot (n - (m - 1)) \wedge n = N\}$
    $q := q + n;$
    $\{q = n \cdot (n - (m - 1)) \wedge n = N\}$
    $m := m - 1$
$\{INV \wedge m = 0\}$
$\{q = N^2\}$

# Esempio: Somma 1..N

Dimostriamo che

$$
\begin{array}{l}
\{n = N \geq 0\} \\
s := 0; \\
\textsf{while } n \neq 0 \textsf{ do} \\
\qquad s := s + n; \\
\qquad n := n - 1 \\
\{s = \sum_{0<I\leq N} I\}
\end{array}
$$

# Soluzione

$\{n = N \geq 0\}$
$\{0 = \sum_{n<I\leq N} I \wedge n \leq N\}$
$s := 0;$
$\{INV : \ s = \sum_{n<I\leq N} I \wedge n \leq N\}$
while $n \neq 0$ do
    $\{INV \wedge n \neq 0\}$
    $\{s + n = \sum_{(n-1)<I\leq N} I \wedge (n-1) \leq N\}$
    $s := s + n;$
    $\{s = \sum_{(n-1)<I\leq N} I \wedge (n-1) \leq N\}$
    $n := n - 1$
$\{INV \wedge n = 0\}$
$\{s = \sum_{0<I\leq N} I\}$

# Esempio: Potenze

Dimostriamo che

$$\{a = A > 0 \wedge b = B \geq 0\}$$
$$r := 1;$$
$$\text{while } b \neq 0 \text{ do}$$
$$\quad r := r \cdot a;$$
$$\quad b := b - 1$$
$$\{r = A^B\}$$

# Soluzione

$\{a = A > 0 \wedge b = B \geq 0\}$
$\{1 = A^{B-b} \wedge a = A > 0 \wedge b \geq 0\}$
$r := 1;$
$\{INV : \ r = A^{B-b} \wedge a = A > 0 \wedge b \geq 0\}$
while $b \neq 0$ do
    $\{INV \wedge b \neq 0\}$
    $\{ra = A^{B-(b-1)} \wedge a = A > 0 \wedge (b-1) \geq 0\}$
    $r := r \cdot a;$
    $\{r = A^{B-(b-1)} \wedge a = A > 0 \wedge (b-1) \geq 0\}$
    $b := b - 1$
$\{INV \wedge \neg(b \neq 0)\}$
$\{r = A^B\}$

# Potenze, Più Efficiente

Algoritmo del contadino russo:

$\{a = A > 0 \wedge b = B \geq 0\}$
$r := 1;$
while $b \neq 0$ do
    if $b$ pari then
        $a := a * a;$
        $b := \lfloor b/2 \rfloor$
    else
        $r := r * a;$
        $b := b - 1$
$\{r = A^B\}$

# Esempio

Bozza di dimostrazione di correttezza:

$\{a = A > 0 \wedge b = B \geq 0\}$
$r := 1;$
$\{INV : \ r \cdot a^b = A^B \wedge a > 0 \wedge b \geq 0\}$
while $b \neq 0$ do
    if $b$ pari then
        $\{b \text{ pari } \wedge INV \wedge b \neq 0\}$
        $\{r \cdot (a \cdot a)^{\lfloor b/2 \rfloor} = A^B \wedge a \cdot a > 0 \wedge \lfloor b/2 \rfloor \geq 0\}$
        $a := a * a;$
        $b := \lfloor b/2 \rfloor$
    else
        $\{b \text{ dispari } \wedge INV \wedge b \neq 0\}$
        $\{(r \cdot a) \cdot a^{b-1} = A^B \wedge a > 0 \wedge b - 1 \geq 0\}$
        $r := r * a;$
        $b := b - 1$
$\{INV \wedge b = 0\}$
$\{r = A^B\}$

# Esercizio

Scrivete un programma per calcolare il fattoriale $n!$ e dimostratene la correttezza.

# Esercizio

Dimostrate che

$$
\begin{array}{l}
\{P\} \\
x := 5; \\
\mathsf{while}\ x \neq 0\ \mathsf{do} \\
\qquad x := x + 1 \\
\{Q\}
\end{array}
$$

per ogni $P, Q$.

Suggerimento: basta farlo per $P = \mathsf{vero}, Q = \mathsf{falso}$ e usare $PrePost$.

# Esempio: Ricerca Lineare

**Problema della ricerca.**
Vogliamo verificare con un programma se un dato intero $Y$ è presente o meno tra i seguenti interi:

$$f(A), f(A+1), f(A+2), \ldots, f(B-2), f(B-1)$$

Inoltre, se presente, vorrei sapere dove si trova nella sequenza una delle occorrenze di $Y$.

Proviamo a formalizzare la specifica di sopra con le triple di Hoare.

# Esempio: Ricerca Lineare

Una possibile specifica formale è:

$$\{a = A \leq b = B \wedge y = Y\}$$
$$c$$
$$\{(\exists Z \in [A..B).f(Z) = Y) \iff p \in [A..B) \wedge f(p) = Y\}$$

La precondizione è semplice da capire.

La postcondizione afferma che, dopo avere eseguito $c$, il valore ricercato $Y$ è presente se e solo se si trova in posizione $p$. Quindi il programma $c$ effettivamente calcola se e dove è presente.

# Esempio: Ricerca Lineare

Nella postcondizione, l'implicazione $\Leftarrow$ è banalmente sempre vera (basta prendere $Z$ uguale al valore di $p$), quindi possiamo semplificare la specifica come segue:

$$\{a = A \leq b = B \wedge y = Y\}$$
$$c$$
$$\{(\exists Z \in [A..B).f(Z) = Y) \implies p \in [A..B) \wedge f(p) = Y\}$$

# Esempio: Ricerca Lineare

Supponiamo di avere esteso IMP con l'espressione $f(e)$, con semantica ovvia.
Un possibile programma che realizza la specifica è:

$\{a = A \leq b = B \wedge y = Y\}$
$p := a;$
while $p < b \wedge f(p) \neq y$ do
$\quad p := p + 1$
$\{(\exists Z \in [A..B).f(Z) = Y) \implies p \in [A..B) \wedge f(p) = Y\}$

Questo algoritmo viene chiamato *ricerca lineare*.
È immediato vedere che ha complessità $O(B - A)$.

# Esempio: Ricerca Lineare

Qui trovare un'invariante richiede di formalizzare che:

- $b, y$ hanno sempre i loro valori $B, Y$.
- $p$ è dentro l'intervallo di ricerca, o al più subito dopo.
- in tutte le posizioni precedenti a $p$ non si trova $Y$.

In formule:

$$INV: \quad b = B \wedge y = Y \wedge A \leq p \leq B \wedge (\forall i \in [A, p).\ f(i) \neq Y)$$

# Soluzione

$\{a = A \leq b = B \wedge y = Y\}$
$\{b = B \wedge y = Y \wedge A \leq a \leq B \wedge (\forall i \in [A, a).\ f(i) \neq Y)\}$
$p := a;$
$\{INV\}$
while $p < b \wedge f(p) \neq y$ do
$\{INV \wedge p < b \wedge f(p) \neq y\}$
$\{b = B \wedge y = Y \wedge A \leq p + 1 \leq B \wedge$
$(\forall i \in [A, p + 1).\ f(i) \neq Y)\}$
$p := p + 1$
$\{INV \wedge \neg(p < b \wedge f(p) \neq y)\}$
$\{(\exists Z \in [A..B).f(Z) = Y) \implies p \in [A..B) \wedge f(p) = Y\}$

# Esercizio

**Esercizio.** Verificare le PrePost della soluzione precedente.

# Ricerca Binaria

Supponendo la monotonia (debole) di $f$

$$\forall a, b.\ a \leq b \implies f(a) \leq f(b)$$

è possibile effettuare un ricerca di un dato $Y$ usando un algoritmo molto più efficiente della ricerca lineare: la *ricerca binaria*.

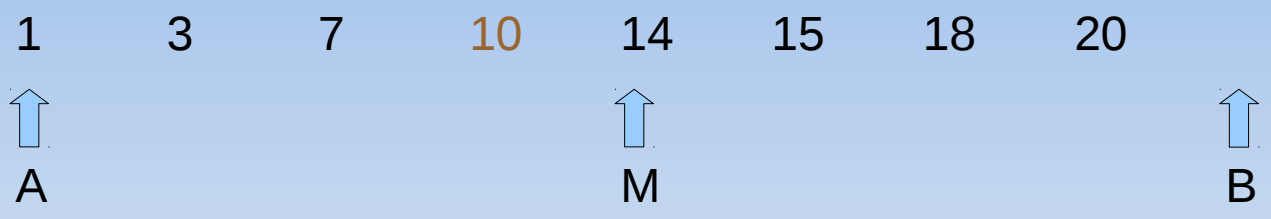
# Ricerca Binaria

Questo algoritmo, informalmente, divide l'intervallo di ricerca in due parti $[A, B) = [A, M) \cup [M, B)$ dove la lunghezza dei due nuovi intervalli  uguale (o al più differisce di un solo elemento).

Valutando quindi se $f(M) < Y$, è possibile restringere la ricerca a solo uno dei due sottointervalli.

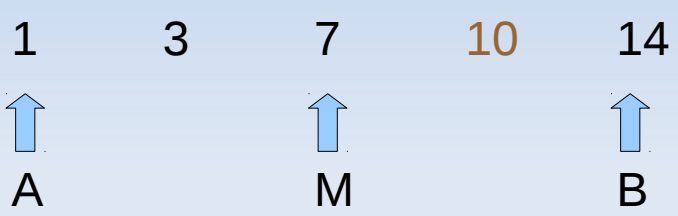
Ripetendo la restrizione, prima o poi si trova $Y$, quando presente.

# Ricerca Binaria

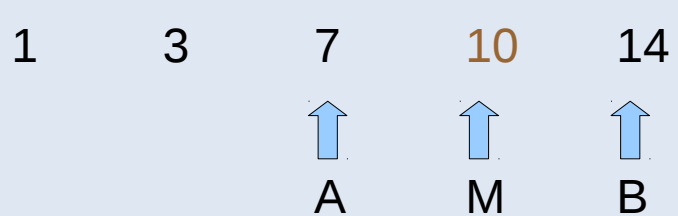



$Y = 10$

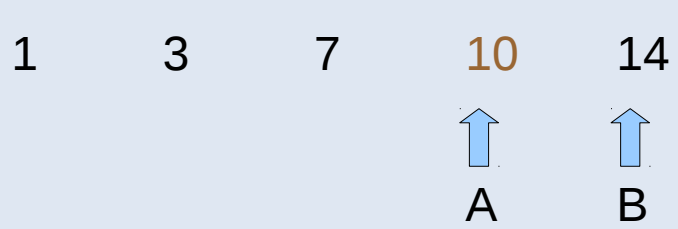
$f(M) > Y$




$f(M) \leq Y$




$f(M) \leq Y$




Nota: $Y$ sta sempre nell'intervallo $[A, B)$

# Ricerca Binaria

Una possibile implementazione:

$\{a = A \leq b = B \wedge y = Y\}$
while $b - a > 1$ do
    $m := \lfloor (a + b)/2 \rfloor$;
    if $f(m) > y$ then
        $b := m$
    else
        $a := m$;
$p := a$
$\{(\exists Z \in [A..B).f(Z) = Y) \implies p \in [A..B) \wedge f(p) = Y\}$

La complessità è $O(\log(B - A))$.

# Esercizi

**Esercizio.** Descrivete informalmente che cosa succede se uso la ricerca binaria per cercare un $y$ che è minore di tutti i valori $f(A), \ldots, f(B-1)$ ? Quali sono i valori delle variabili alla fine?

**Esercizio.** Più in generale, che cosa succede se $y$ è diverso da tutti i valori $f(A), \ldots, f(B-1)$ ? Ci sono tre casi: $y$ è minore di tutti, $y$ è maggiore di tutti, o . . . In ogni caso, quali sono i valori delle variabili alla fine?

**Esercizio.** (impegnativo) Provate a dare un'invariante per la ricerca binaria, e a dimostrarne la correttezza.