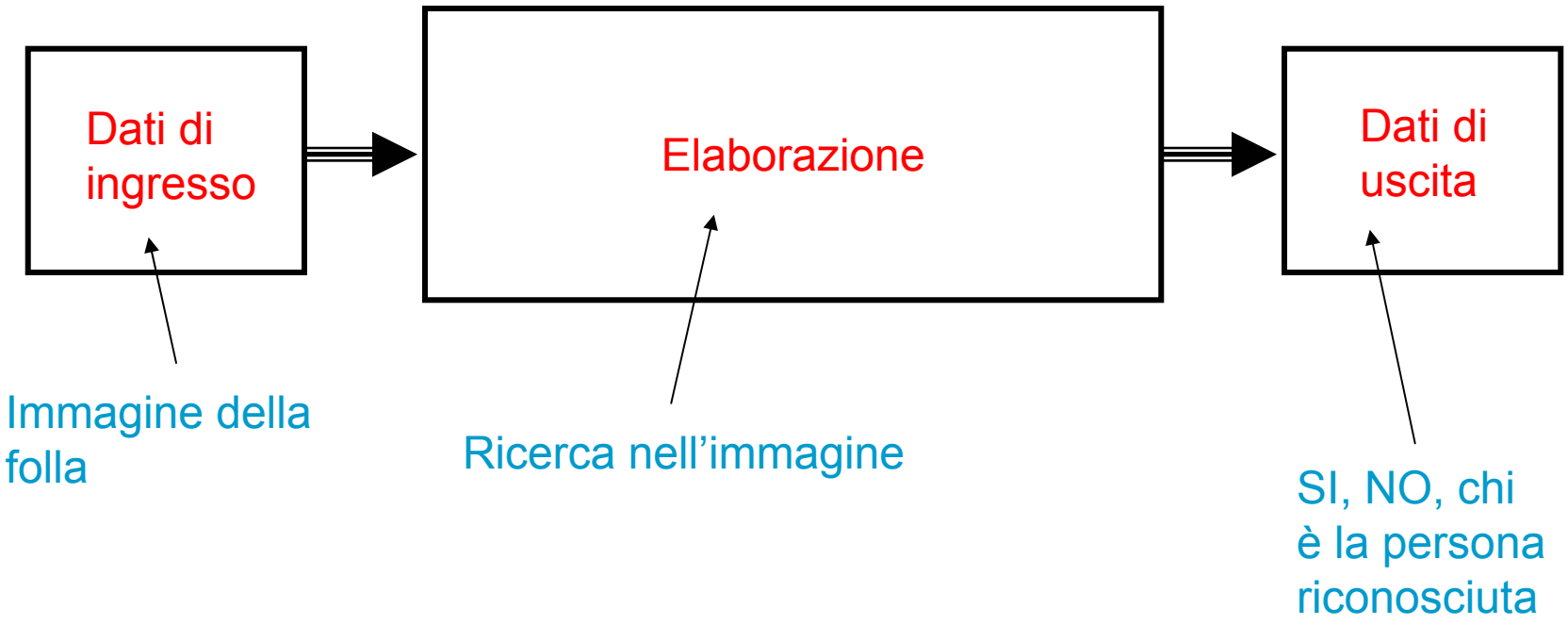
Problemi, Algoritmi e Programmi

Approfondiamo come *progettare e scrivere* nuovi algoritmi e programmi

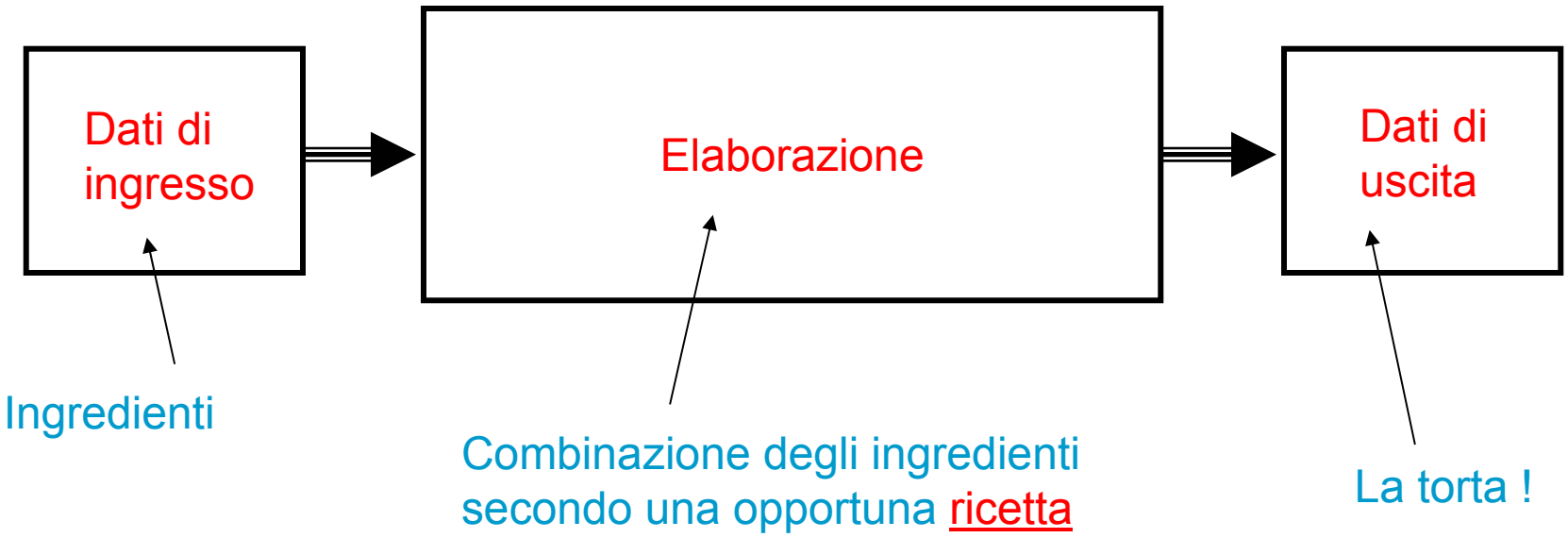
Risolvere un problema

- es : riconoscere qualcuno fra la folla



Risolvere un problema

- es : torta di carote

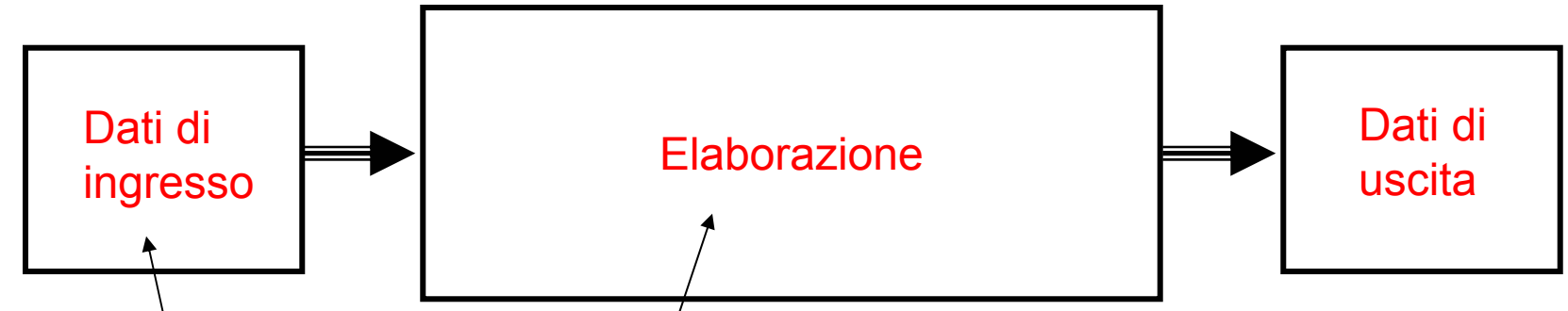


Risolvere un problema

– vogliamo essere capaci di specificare la strategia seguita dal passo di elaborazione in modo da farla eseguire ‘automaticamente’ dal computer quindi dobbiamo :

– *riuscire a descrivere accuratamente i vari passi della soluzione attraverso azioni che il calcolatore è in grado di effettuare e con un linguaggio che è in grado di comprendere*

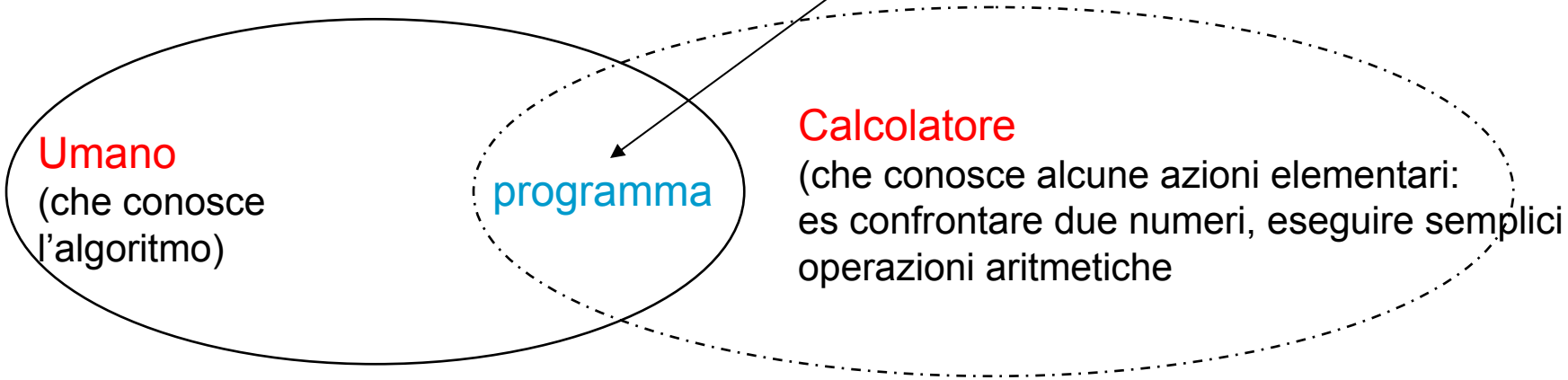
Risolvere un problema



Codificati opportunamente

Trasformazione dei dati di ingresso seguendo i passi specificati da un opportuno algoritmo

Ovvero la descrizione dell'algoritmo secondo un linguaggio comprensibile al calcolatore



Qual'è il ruolo dei calcolatori ?

- Nel loro impiego tradizionale, i calcolatori sono essenzialmente esecutori di soluzioni che esseri umani hanno previamente identificato e descritto
- Questo utilizzo è motivato dalla notevole velocità di esecuzione dei calcolatori e dalla loro capacità di eseguire molte volte la stessa operazione
- Un calcolatore è caratterizzato
  - dal linguaggio che è in grado di interpretare e
  - dalle istruzioni che è in grado di eseguire

Introduzione alla programmazione

- Prima di scrivere un programma:
  - Avere una piena comprensione del problema
  - Pianificare con cura un approccio per risolverlo
- Mentre si scrive un programma:
  - Sapere quali “mattoni per costruire” sono disponibili
  - Seguire buoni principi di programmazione

Algoritmi

- Problemi di elaborazione
  - Possono essere risolti eseguendo, in un ordine specifico, una serie di azioni
- Algoritmo: procedura in termini di
  - Azioni che devono essere eseguite
  - L'ordine in cui tali azioni devono essere eseguite

Algoritmi e Programmi

- Algoritmo (def) :
  - una sequenza di azioni *non ambigue* che trasforma i dati iniziali nel risultato finale utilizzando un insieme di azioni elementari che possono essere eseguite da un opportuno esecutore.
- Programma (def)
  - specifica di un algoritmo utilizzando un linguaggio non ambiguo e direttamente comprensibile dal computer

Pseudocodice

– Linguaggio artificiale e informale, che aiuta i programmatori a sviluppare gli algoritmi

– Simile all'italiano di tutti i giorni

– Non realmente eseguito sui computer

– Aiuta il programmatore a "riflettere" sul programma, prima che provi a scriverlo

- Facilmente convertibile in un corrispondente programma C

Strutture di controllo

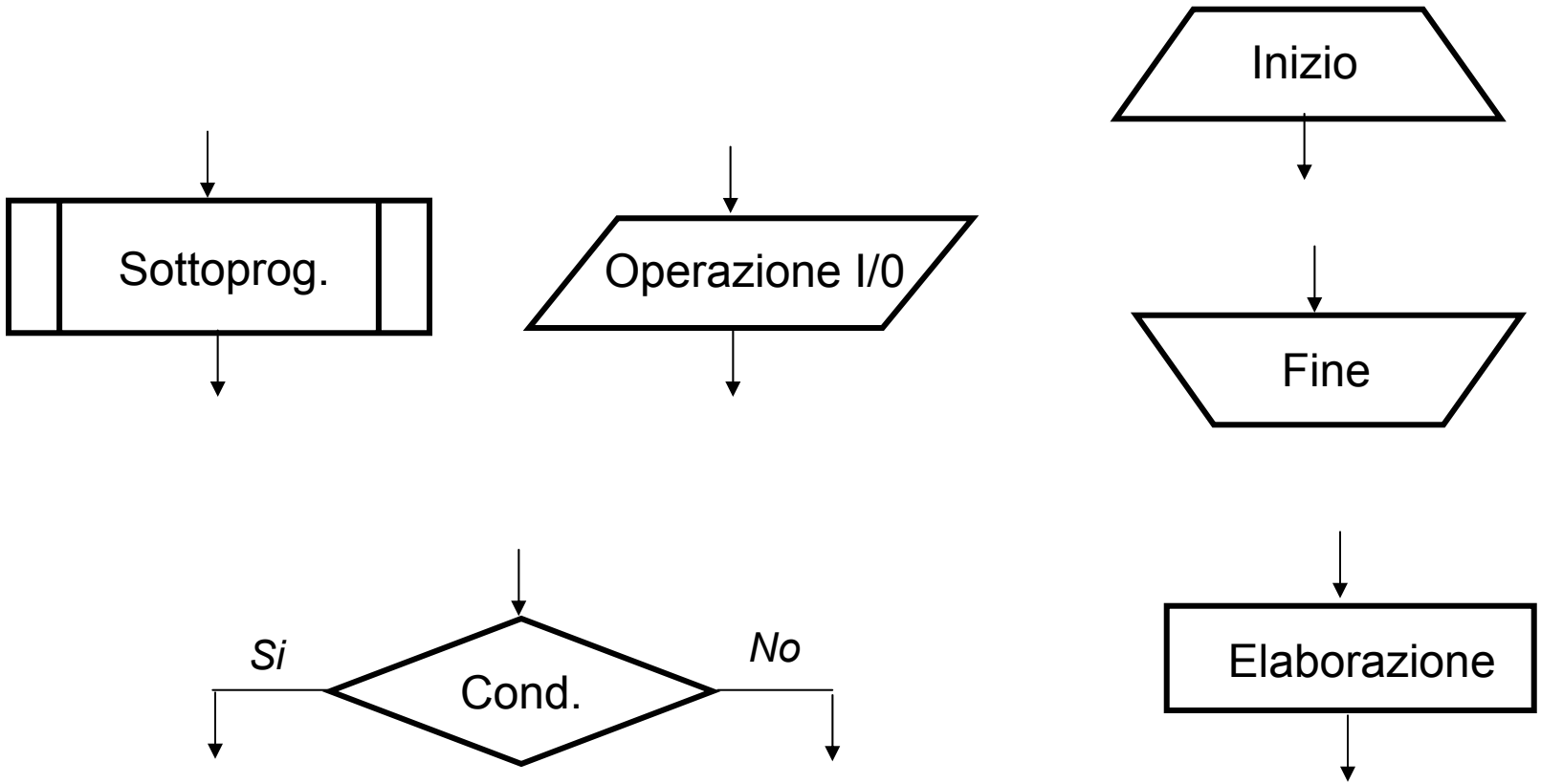
- Esecuzione sequenziale
  - Le istruzioni sono eseguite, una dopo l'altra, nell'ordine in cui sono state scritte
- Trasferimento di controllo
  - Quando la prossima istruzione ad essere eseguita non è la prossima nella sequenza
- Strutture di controllo
  - Tutti i programmi possono essere scritti in termini di tre sole strutture di controllo:
    - *Struttura di sequenza*: le istruzioni vengono eseguite sequenzialmente in modo implicito
    - *Struttura di selezione*: Se, Se…altrimenti
    - *Struttura di iterazione*: Finché

Diagramma di flusso

- Sono grafici che permettono di esprimere un algoritmo in modo preciso ed intuitivo
- Si costruiscono a partire da un certo numero di ‘blocchi base’ che rappresentano le operazioni elementari ed i costrutti di controllo

Diagramma di flusso

I blocchi base:



Il comando di selezione Se

- Struttura di selezione:
  – Usata per scegliere tra percorsi di azione alternativi
  – Pseudocodice:

Se il voto dello studente è maggiore o uguale a 60
Visualizza "Promosso"

- Se la condizione è vera
  – Sarà visualizzato "Promosso" ed "eseguita" l'istruzione sucessiva
  – Se falsa, la visualizzazione sarà ignorata e sarà eseguita l'istruzione successiva
  – I rientri rendono i programmi più semplici da leggere

Il comando di selezione Se

Simbolo rombo (simbolo di decisione)

– Indica che dovrà essere eseguita una scelta

– Contiene un’espressione che può essere vera o falsa

– Testa la condizione, segue il percorso appropriato

Il comando di selezione Se … altrimenti

- **Se**

– Esegue l'azione indicata solo quando la condizione è vera

- **Se…altrimenti**

– Specifica che, nel caso in cui la condizione sia vera, dovrà essere eseguita una azione differente da quella che si dovrà eseguire qualora la condizione sia falsa

- **Pseudocodice**:

Se il voto dello studente è maggiore o uguale a 60
Visualizza "Promosso"
altrimenti
Visualizza "Bocciato"

– Osservate le convenzioni di rientro e spaziatura

Il comando di selezione Se … altrimenti

- Diagramma di flusso del comando di selezione

I comandi Se … altrimenti nidificati

Se il voto dello studente è maggiore o uguale a 90
Visualizza “A”
altrimenti
Se il voto dello studente è maggiore o uguale a 80
Visualizza “B”
altrimenti
Se il voto dello studente è maggiore o uguale a 70
Visualizza “C”
altrimenti
Se il voto dello studente è maggiore o uguale a 60
Visualizza “D”
altrimenti
Visualizza “F”

Diagramma di flusso

- I blocchi base vengono collegati tramite ‘frecce’ che collegano un’azione alla successiva all’interno dell’algoritmo
- Vediamo il diagramma di flusso del seguente algoritmo:

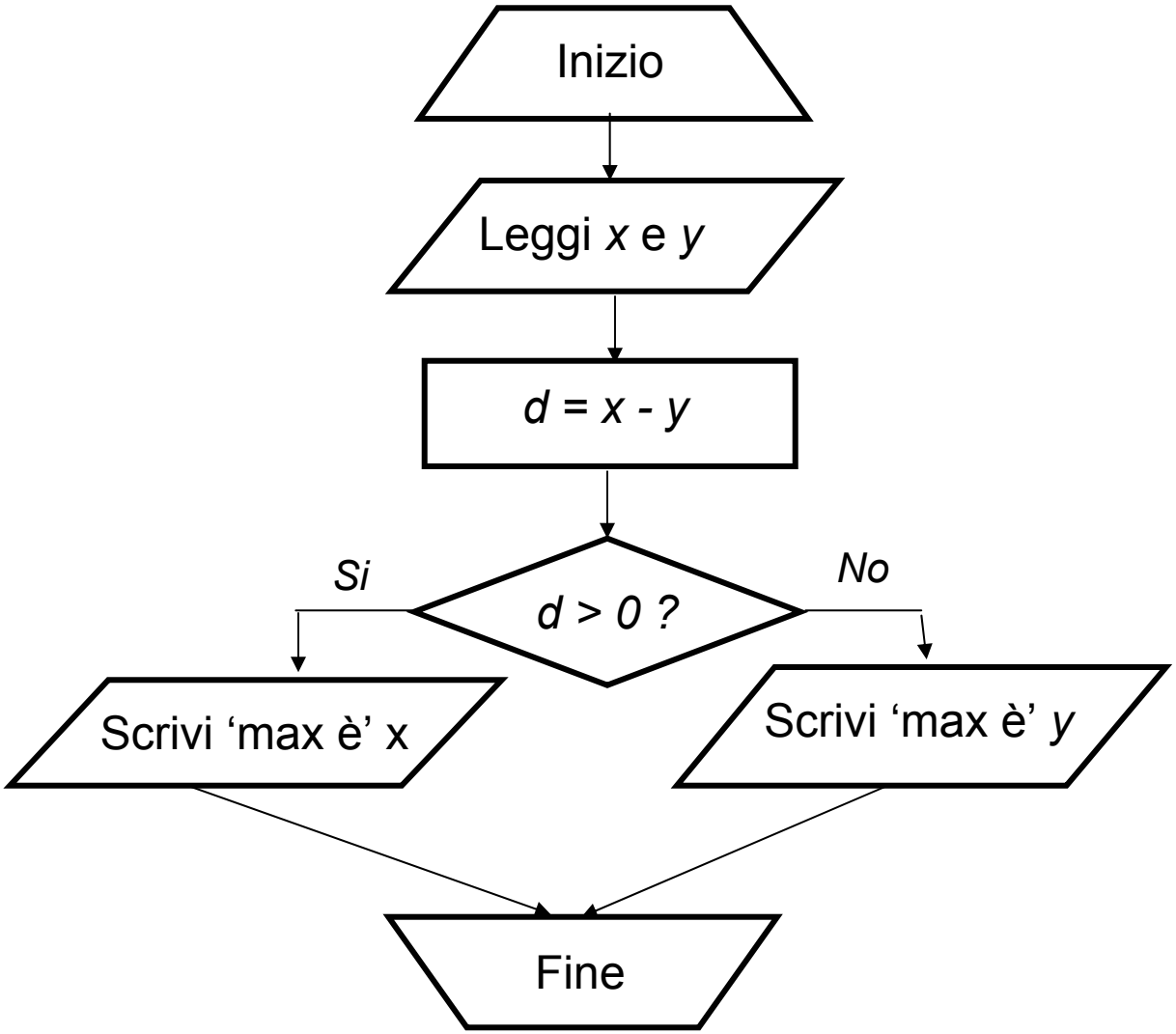
Trovare il maggiore fra 2 numeri interi *x* e *y*

Il maggiore fra due numeri interi *x, y*

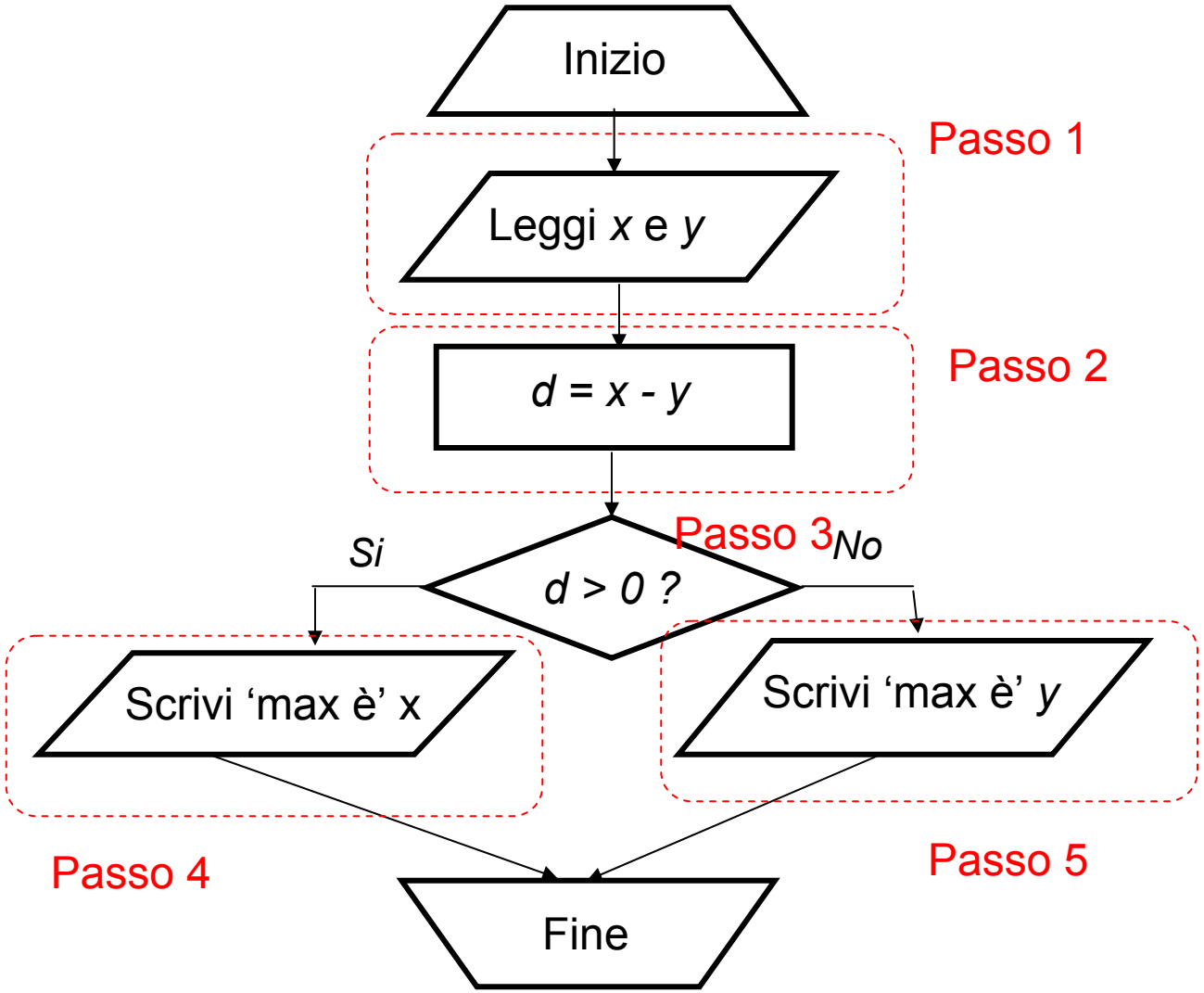
Algoritmo max

1. Leggi i valori di *x* e *y* dall'esterno
2. Calcola la differenza *d* fra *x e y (d=x-y)*
3. Se d è maggiore di 0 allora esegui il passo 4 altrimenti esegui il passo 5
4. Stampa 'il massimo è …' seguito dal valore di x e termina
5. Stampa 'il massimo è …' seguito dal valore di y e termina

DF di max



DF di max


Inizio
Leggi x e y
Passo 1
d = x - y
Passo 2
Passo 3
Si
No
d > 0 ?
Scrivi 'max è' x
Scrivi 'max è' y
Passo 4
Passo 5
Fine

DF e programmi

- I DF sono un primo passo verso la formalizzazione di un algoritmo in modo non ambiguo
- Per ottenere una codifica interpretabile direttamente dalla macchina dobbiamo però specificare molti più dettagli :
  - trasformare tutte le ‘frasi’ in variabili e modifiche su di esse

Programmi

- Per ottenere una codifica interpretabile direttamente dalla macchina dobbiamo anche :
  - decidere come codificare l'informazione
    - non banale in esempi più complessi del nostro, ad esempio se voglio codificare un'immagine o un video
  - scrivere il tutto con una codifica 'leggibile' dalla macchina
    - … ma la macchina lavora molto a basso livello (linguaggio macchina, codificato con zeri e uni) !!!!

Programmi

- … soluzione….
  - usare linguaggi di 'livello' più alto (*linguaggi di programmazione ad alto livello*)
  - usare dei programmi appositi per far tradurre i nostri programmi in linguaggio macchina (i *compilatori*)
- importante ….
  - I tipici linguaggi (C, C++, Java, Fortran, Basic…) permettono di definire strutture del tutto analoghe ai diagrammi di flusso che abbiamo visto finora

Programma: ***max*** in C

```
main() /* calcola max */
{
int x, y, d;

scanf ("%d %d”, &x, &y) ;
d = x - y ;
if (d > 0)
   printf (”il max è %d”, &x) ;
else
   printf (”il max è %d”, &y) ;
return ;
}
```

Comando: if-else

La forma generale dell'istruzione if-else è la seguente:

```
if (espressione)
        istruzione;
else
        istruzione;
```

dove istruzione può essere una singola istruzione, un blocco di istruzioni o l'istruzione nulla. La clausola else è opzionale.

- Se espressione fornisce un risultato vero, viene eseguita l'istruzione o il blocco relativo alla parte if;
- Se espressione fornisce un risultato falso, verrà eseguita, se esiste, l'istruzione o il blocco else.

Comando: if-else

```
/* Scrivere un programma che legge due numeri e stampa il maggiore */

#include <stdio.h>
    int main ( ) {
    int x, y, max;
    printf (“Digita due numeri: ”);
    scanf (“%d%d”, &x, &y);
    if (x>y)
            max = x;
    else
            max = y;
    printf (“%d\n”, max);
}
```

Comando: if-else

```
/* Scrivere un programma come prima che non usa la variabile max */

#include <stdio.h>
    int main ( )
   {
    int x, y;
    printf (“Digita due numeri: ”);
    scanf (“%d%d”, &x, &y);
    if (x>y)
            printf (“%d\n”, x);
    else
           printf (“%d\n”, y);
}
```

Comando: if-else

/* Scrivere un programma che riceve un numero intero in input, determina se il numero e' maggiore o minore di 100 e stampa a video un messaggio corrispondente */

```
#include <stdio.h>
int main( )  {
    int i;                                  /* dichiarazione di variabile */
    printf ("Dammi un intero: ");
    scanf ("%d", &i);                       /* inizializzazione della variabile i */
    if (i<100)
        printf ("il numero inserito è minore di 100\n");
    else
        printf ("il numero inserito è maggiore o uguale a 100\n");   }
```

Espressioni booleane

```
/* Utilizzo delle espressioni booleane */

#include <stdio.h>
int main( )
 {
    int a, i;
    printf (“Dammi un intero: ”);
    scanf (“%d”, &i);
    a = i<100;
    if (a!=0)                                   /* equivale a if (i<100) */
          printf (“il numero inserito è minore di 100\n”);
    else
          printf (“il numero inserito è maggiore o uguale a 100\n”);
}
```

Espressioni booleane

L'assegnamento

a=i<100

è del tutto lecito, perché viene valutata l'espressione logica **i<100**, che può restituire **1** (**true**) o **0** (**false**).
Il risultato e' dunque un numero intero, che viene assegnato alla variabile, di tipo int, **a**.

Valutare l'espressione

a!=0

significa chiedersi se il valore di **a** è diverso da **0**. Ma questo equivale a chiedersi se il valore di **a** è **true**.
Si può dunque modificare il programma precedente come segue.

Espressioni booleane

```
/* Un altro esempio di utilizzo delle espressioni booleane */

#include <stdio.h>
int main( )
 {
    int a, i;
    printf (“Dammi un intero: ”);
    scanf (“%d”, &i);
    a = i<100;
    if (a)                                   /* equivale a if (i<100) */
          printf (“il numero inserito è minore di 100\n”);
    else
          printf (“il numero inserito è maggiore o uguale a 100\n”);
}
```

Espressioni booleane

Dato che le espressioni booleane restituiscono un risultato numerico, non esistono differenze tra le espressioni booleane e quelle aritmetiche.

Un'espressione può contenere una combinazione di operatori aritmetici, logici e relazionali.

Esempio

```
w = k + numero + (i < 100);
```

l'assegnamento viene effettuato alla fine, perché = ha la priorità più bassa.

Attenzione

Essendo l'operatore di assegnamento trattato alla stregua degli altri, sarà lecita anche la seguente espressione:

i > n && (x=y)

dove **i > n** è vera se il valore di i è maggiore di n;
mentre **x=y** corrisponde all'assegnamento di **y** alla variabile **x**:
in questo caso, se il valore di **y** è diverso da zero l'espressione **(x=y)** risulta vera altrimenti è falsa.

Queste caratteristiche rendono il C un linguaggio flessibile che consente la scrittura di codice sintetico ed efficiente ma anche difficilmente interpretabile.