# Programma del corso

- *Introduzione agli algoritmi*
- *Rappresentazione delle Informazioni*
- *Architettura del calcolatore*
- *Reti di Calcolatori*
- ***Elementi di Programmazione***

# Algoritmi e programmi

- **Algoritmo**

Sequenza finita di passi che risolve in tempo finito un problema.

- **Codifica**

Fase di scrittura di un algoritmo attraverso un insieme ordinato di frasi (“istruzioni”), scritte in un qualche **linguaggio di programmazione**, che specificano le azioni da compiere.

## Programma

Testo scritto in accordo con la sintassi e la semantica di un linguaggio di programmazione.

# Cenni di programmazione

- Il computer esegue programmi
- Un programma eseguibile dal computer è una sequenza di istruzioni comprensibili da quel computer
- Usando sequenze diverse di istruzioni, e dati diversi, possiamo far fare al computer le cose più disparate

# Cenni di programmazione

- Il linguaggio di programmazione che viene compreso da un calcolatore è il linguaggio macchina (**assembly**)
- Però scrivere programmi in assembly è scomodo, perché il linguaggio è molto distante da quello umano.
- Inoltre, un programma in assembly gira solo su un tipo di cpu
  - sarebbe comodo poter usare lo stesso programma su cpu diverse senza doverlo riscrivere ogni volta

# Linguaggi di programmazione ad alto livello

- I linguaggi di programmazione ad alto livello permettono di scrivere programmi con una notazione più vicina a lingue naturali
- Usando degli opportuni traduttori (compilatori ed interpreti) lo stesso programma può essere usato su macchine diverse
- Esempi: C, C++, Java, Perl, Python, Basic, Lisp, Fortran, Cobol, Pascal, Ada, ML, Prolog,...

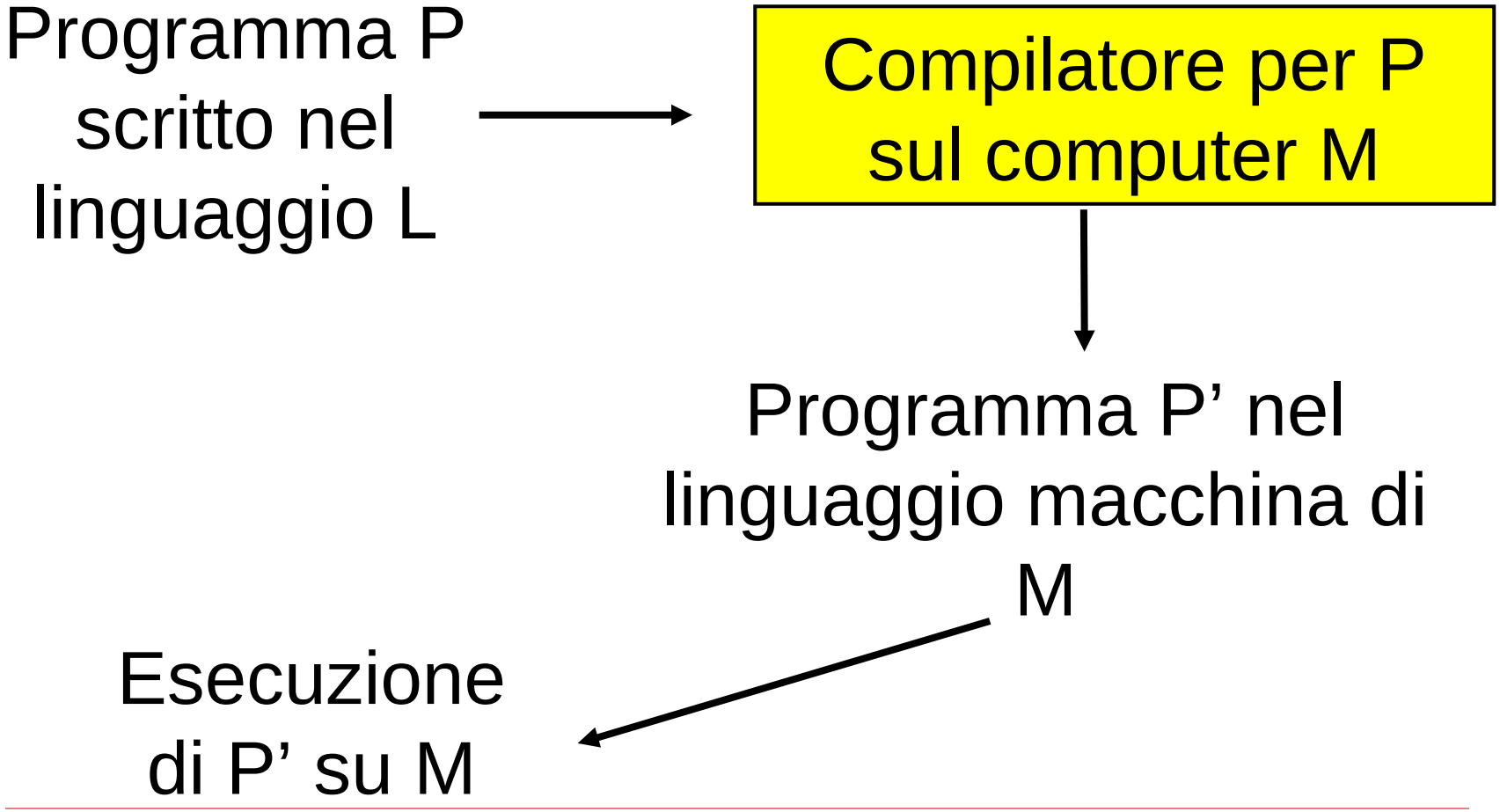
# Linguaggi di programmazione ad alto livello

- Un linguaggio artificiale per scrivere programmi
- Un linguaggio preciso e rigoroso. Occorre rispettare:
  - la sintassi (regole di composizione dei simboli del linguaggio per ricavarne le istruzioni)
  - la semantica (significato delle istruzioni)
- Il computer è meno tollerante agli “errori” di un umano

# Il compilatore

- Un programma scritto in un linguaggio ad alto livello è detto **programma sorgente**.
- Per essere eseguito su un computer, va tradotto nel linguaggio macchina del computer.
- Il **compilatore** è un programma che esegue la traduzione, producendo il **programma oggetto**
- Il compilatore segnala anche eventuali errori di sintassi nella scrittura del programma sorgente

# Il compilatore



# L’interprete

- In alternativa alla compilazione, un programma sorgente può essere **interpretato**.
- Un **interprete** è un programma che non produce alcun programma oggetto, ma legge ogni istruzione del programma sorgente e genera “al volo” le istruzioni macchina corrispondenti, che vengono passate all’hardware per l’esecuzione.

# Compilatori vs interpreti

- In un programma compilato, la traduzione avviene una sola volta, e poi il programma oggetto può essere eseguito ripetutamente
- In un programma interpretato, la traduzione avviene tutte le volte che si esegue il programma
- Molti linguaggi permettono entrambe le scelte

# Dall’algoritmo al programma

# Il concetto di algoritmo

- Un algoritmo è una sequenza di passi necessari per risolvere un problema o eseguire una computazione
- In alcuni casi, lo stesso problema/computazione può essere risolto in modi diversi, ai cui corrispondono diversi algoritmi
- Un programma non è altro che  la descrizione di un algoritmo scritta nel linguaggio di programmazione scelto.

# Esempio di algoritmo:

## ricerca di una voce nell'elenco telefonico

- Sia **cognome** la voce da cercare
- Sia **E** l'elenco da "sfogliare"

Ripeti:

- se E è vuoto allora cognome non è nell'elenco, termina.
- prendi la pagina a metà dell'elenco E
- esamina tutte le voci della pagina che hai di fronte
- se trovi cognome allora annota il numero e termina.
- se cognome precede la prima voce della pagina
  - considera come E la prima metà dell'elenco

altrimenti

  - considera come E la seconda metà dell'elenco

Fine

# Diagrammi di flusso

- Notazione grafica usata per descrivere in modo intuitivo le azioni di cui è fatto un algoritmo.
- Viene usata per descrivere i passi salienti di un algoritmo, senza doversi preoccupare dei dettagli sintattici del programma corrispondente
- Una volta che l'algoritmo è stato descritto con un diagramma di flusso, deve però essere trasformato nel programma corrispondente.
- Ogni azione è rappresentata da un **blocco**

# Diagrammi di flusso

- Notazione grafica usata per descrivere in modo intuitivo le azioni di cui è fatto un algoritmo.
- Viene usata per descrivere i passi salienti di un algoritmo, senza doversi preoccupare dei dettagli sintattici del programma corrispondente
- Una volta che l'algoritmo è stato descritto con un diagramma di flusso, deve però essere trasformato nel programma corrispondente.
- Ogni azione è rappresentata da un **blocco**

# Il concetto di variabile

- Quasi tutti i linguaggi ad alto livello mettono a disposizione le variabili: “contenitori” (con un nome di riferimento) per informazioni
- Sono astrazioni di regioni di memoria in cui si trovano informazioni di un specificato tipo
- Tipo di una variabile: il modo in cui sono codificate le informazioni
- Esempi: numero intero (nella codifica del complemento a due), sequenza di caratteri, ...

# Il concetto di variabile

- Quando si scrive un programma è necessario dichiarare quali variabili vogliamo usare.
- Una dichiarazione specifica il nome e il tipo di una variabile, ad esempio:
  - Variabile **LETTERA**, tipo: carattere;
  - Variabile **SOMMA**, tipo: intero;

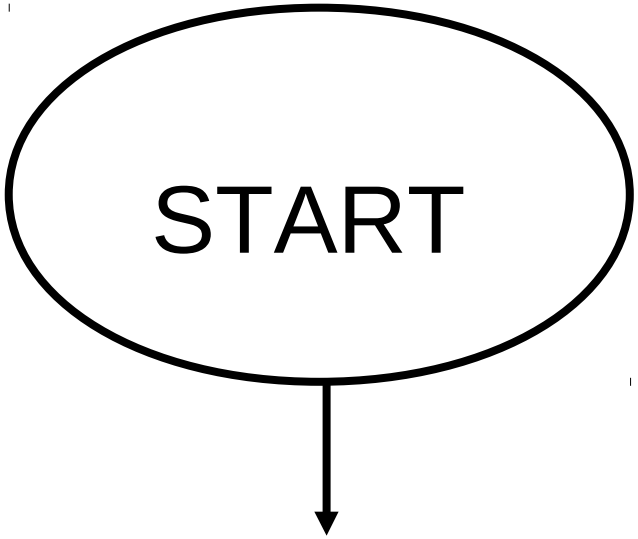
# Il concetto di variabile

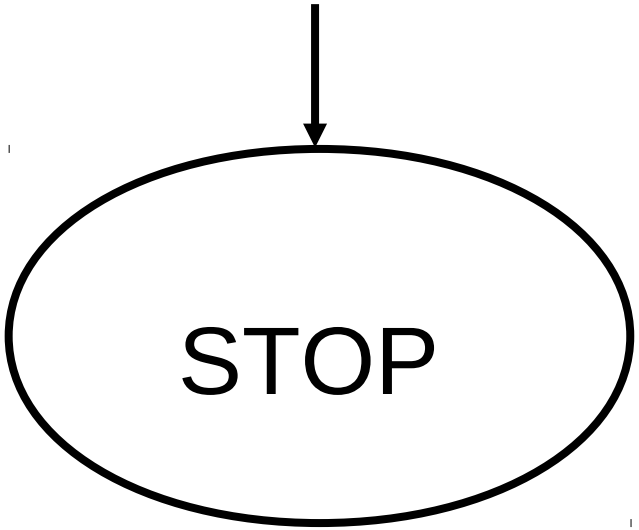
- Ogni variabile ha un nome, che si usa nel programma per riferirsi alla variabile stessa.
- Una variabile contiene un valore (del tipo della variabile) che può essere modificato.
- Durante l'esecuzione di un programma esiste una associazione tra il nome di ogni variabile e la zona di memoria in cui è memorizzato il suo valore.
- Quindi variabili sono una **astrazione di sezioni di memoria**.

# L’importanza delle variabili

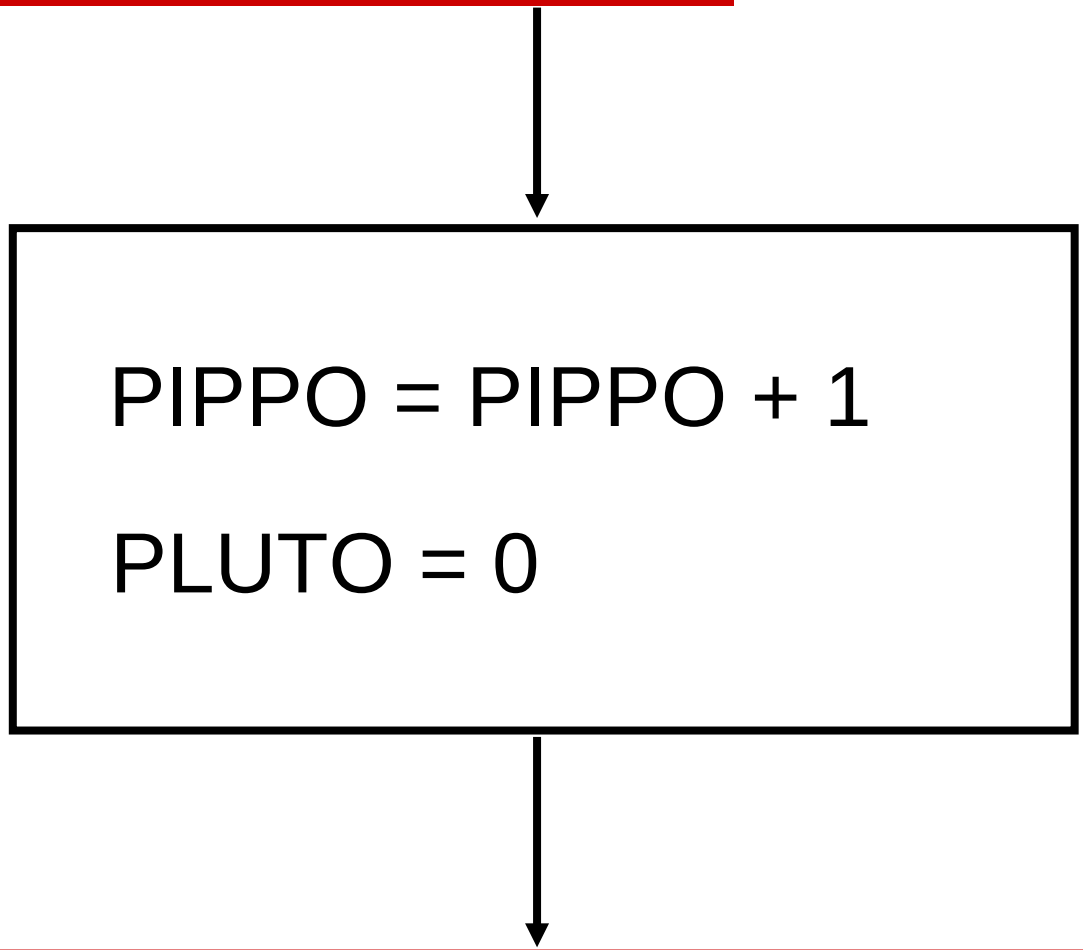
- Le variabili sono lo strumento fondamentale per assicurare la flessibilità dei programmi.
- Lo stesso programma, eseguito con variabili di valore diverso, produce risultati diversi. Lo stesso programma si adatta cioè alle esigenze del momento, senza dover essere riscritto.

# Blocchi di flusso: inizio e fine algoritmo

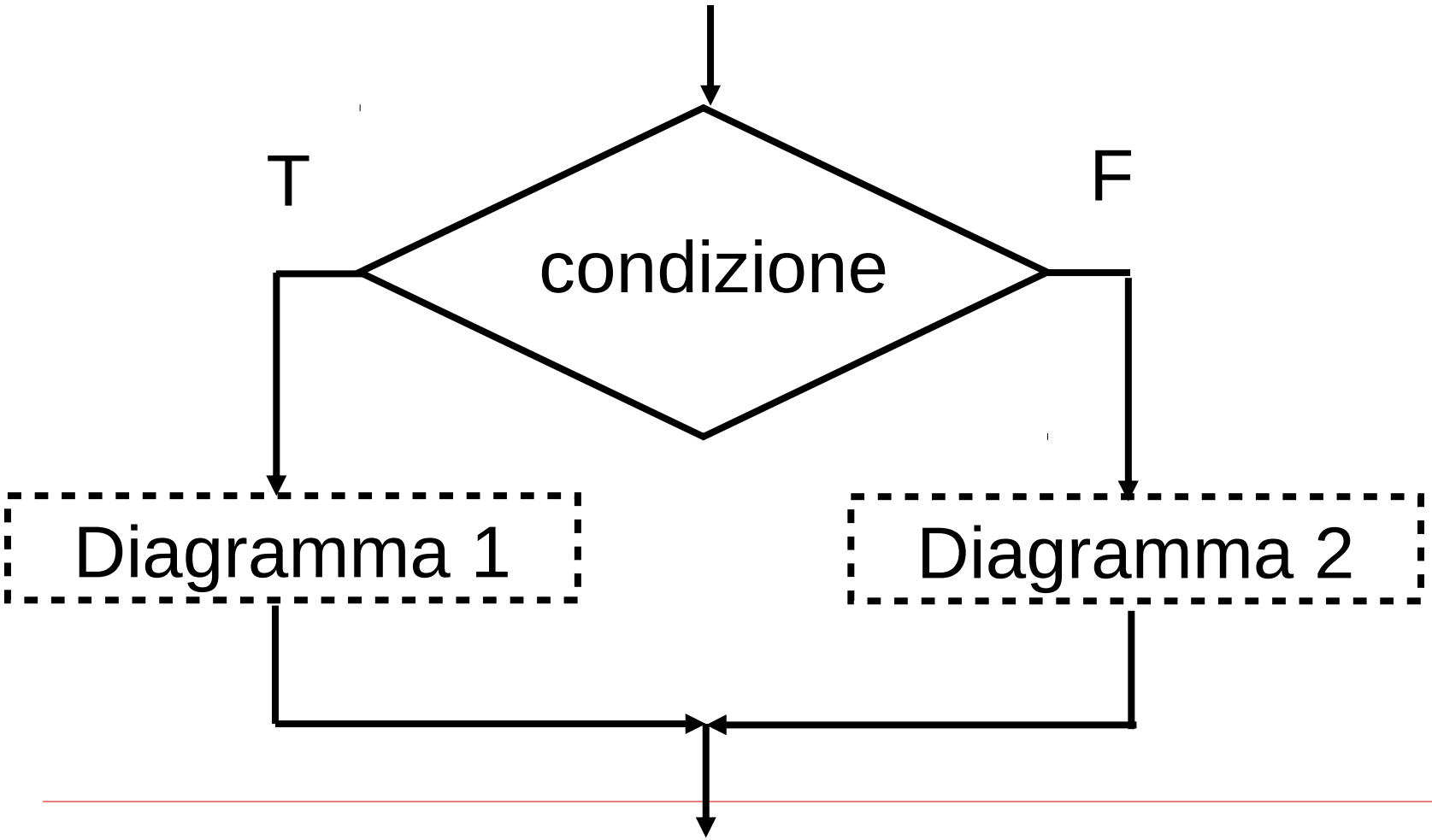


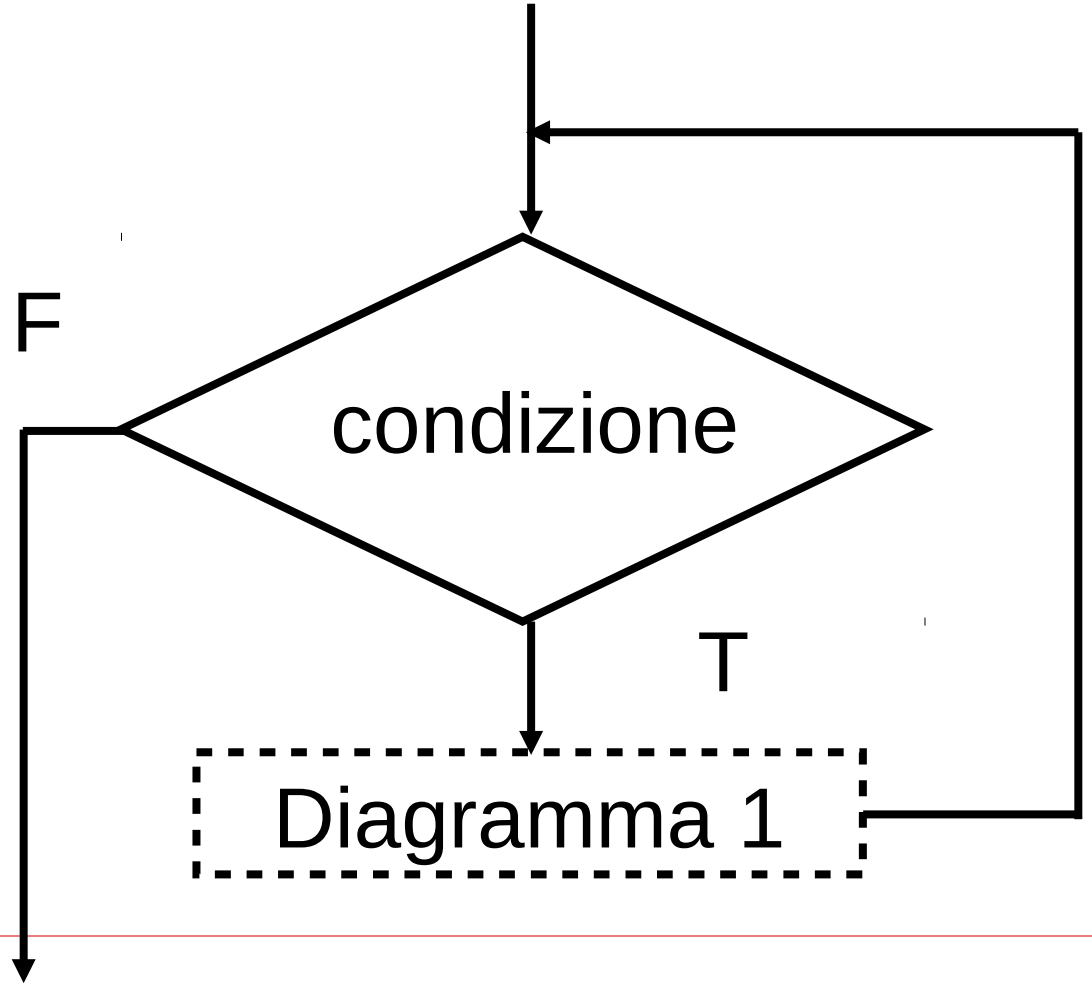
# Blocchi di flusso: una o più azioni elementari



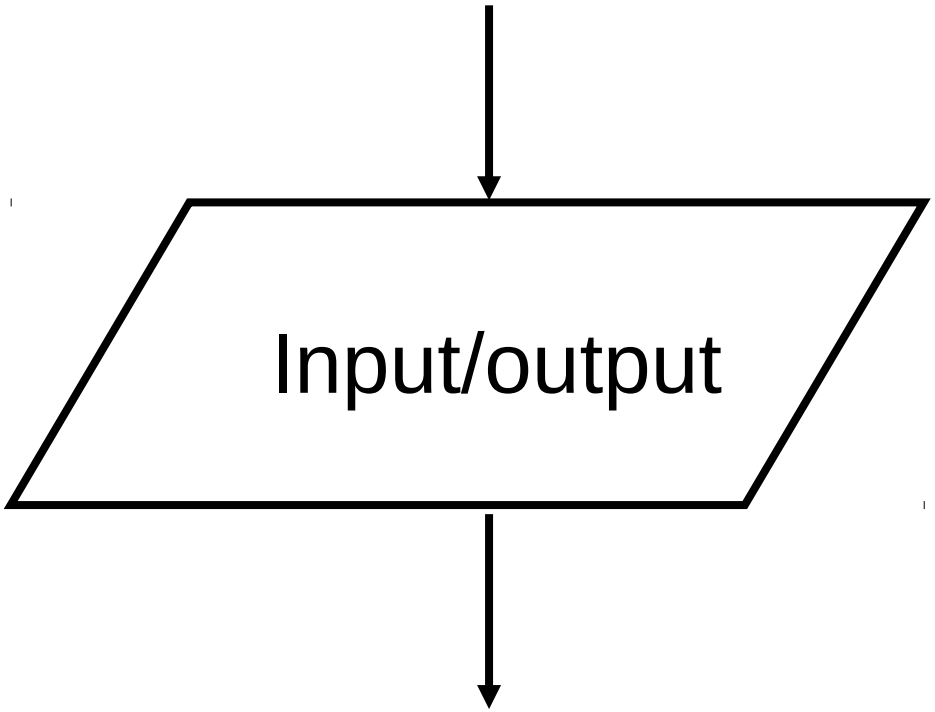
# Blocchi di flusso: Blocco condizionale



# Blocchi di flusso: Blocco di ripetizione



# Blocchi di flusso: Input/Output



# Visual Basic

# Visual Basic

- Linguaggio di programmazione per lo sviluppo di applicazioni da eseguire in un qualunque ambiente Microsoft Windows

- Visual Basic è una variante della Microsoft dell'originario linguaggio Basic

# Visual Basic

- Linguaggio di programmazione
  - Ad alto livello
  - Interpretato (ambiente interattivo)
  - Visuale
  - Guidato dagli eventi

# Processo di programmazione in Visual Basic

- Progetto dell'applicazione
- Creazione dell'interfaccia grafica
- Aggiunta del codice agli elementi visivi
- Esecuzione

# Processo di programmazione in Visual Basic

- Un programma è costituito da
  - Controlli (ciascuno rappresentato da un'icona)
  - Istruzioni
- L'interazione dell'utente con un qualunque controllo rappresenta un evento
- Ad ogni controllo sono associati diversi eventi
- Il codice è suddiviso in blocchi (routine)
- Una routine contiene il codice eseguito quando viene generato un evento su un controllo
  - Click del mouse, pressione di un tasto, apertura di una finestra, ecc.

# Tipi di dati

- **Integer** (valori interi -32768 a 32767)
  - Esempio: 0, 123, -4534
- **Double** (valori reali)
  - Esempio: 0, 12.34, 0,123 E+123
- **String** (sequenza di caratteri scritta tra virgolette)
  - Esempio: “Ciao”, “Inserisci il primo numero”
- **Boolean** (valori true e false)

# Operatori

- Operatori matematici:+, -, *, /, Mod, \
- Operatori su stringhe: + oppure &
  - Esempio: “Visual” & “Basic” = “VisualBasic”
- Operatori condizionali: =, <=, >=, <, >, <>
- Operatori logici: AND, OR, XOR, NOT

# Costrutti di base

- Definizione delle variabili usate nel programma e del loro tipo:

```
Dim SOMMA, N As Integer
Dim NOME As String
Dim RISULTATO As Double
```

# Costrutti di base

- Assegnamento di un valore (può essere un valore costante, il valore di una variabile o combinazioni tramite operatori) ad una variabile (i tipi devono essere sempre essere giusti!):

```
PIPPO = 5
PLUTO = 7
RISULTATO = PIPPO/PLUTO
TESTO = “Ciao”
```

# Costrutti di base

- Ingresso/Uscita

  - **X = InputBox(s)**

    (apre una finestra in cui è scritta la stringa s, e assegna ad X un valore inserito dall'utente)

  - **Print X**

    (stampa X su output)

# Costrutti di base

- Azioni condizionali, esempio:

```
If A*A+B*B = C*C Then
    Print “Triangolo rettangolo”
Else
    Print “Triangolo non rettangolo”
End If
```

# Costrutti di base

- Azioni ripetute, esempio:

```
While (I < 100)
    SOMMA = SOMMA + X
    I = I + 1
Wend
```