Prof. Mario Ciampi

Elementi di Informatica

INGEGNERIA MECCANICA E AEROSPAZIALE

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

Iniziamo a programmare

Programmare

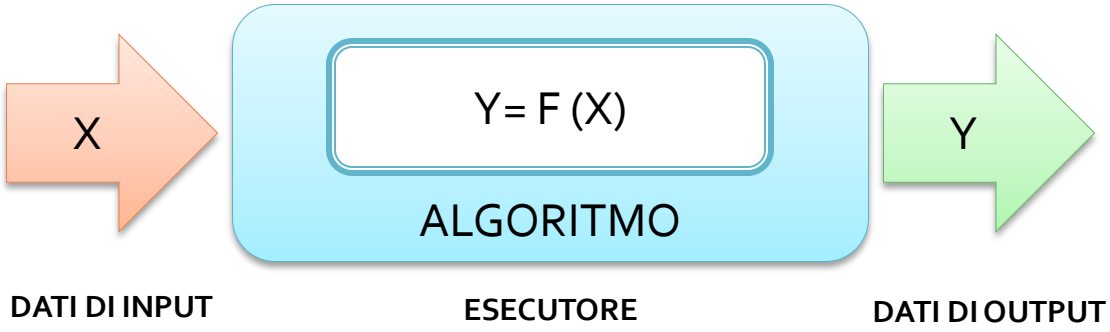
- Programmare
  - Definire un insieme di attività che devono essere svolte secondo un ordine prestabilito
    - Esempi di programmi
      - Libretto di istruzioni
      - Ricetta di cucina
      - Teorema matematico

Programmare

- L'**esecutore** di un programma
  - Soggetto che è in grado di
    - Comprenderlo
    - Eseguirlo
  - **Istruzioni**
    - Frasi del programma che istruiscono l'esecutore sul da farsi
  - **Dati**
    - Oggetti di cui l'esecutore si serve per eseguire il programma

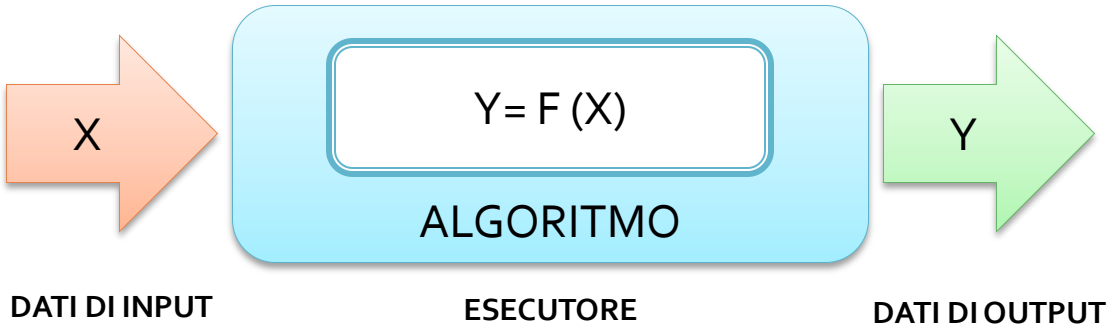
Programmare

- Un qualsiasi programma elabora dati
  - È una *trasformazione* di *dati di ingresso* in *dati di uscita*



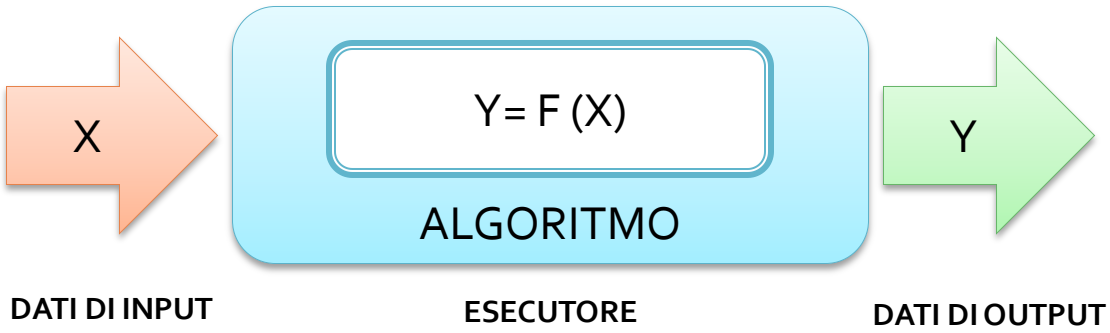
Programmare

- Linguaggio
  - Un programma deve essere espresso in un *linguaggio* che è noto all'esecutore
    - Ma i procedimenti sono indipendenti dal linguaggio



Programmare

- Algoritmo
  - Sequenza finita di passi da eseguire
    - Procedimento la cui validità è indipendente dal linguaggio



Programmare

- Programma
  - Traduzione dell'algoritmo progettato in un linguaggio comprensibile per l'esecutore a cui è destinato
- Programmare
  - Progettare algoritmi indipendentemente dal linguaggio dell'esecutore

Programmare

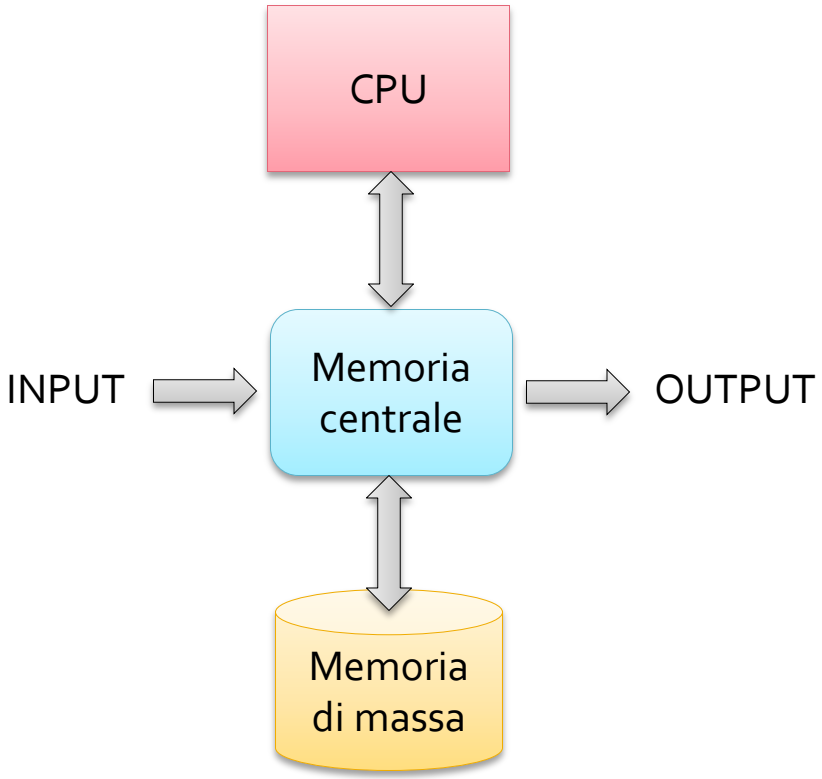
- L'informatica si interessa dello studio degli algoritmi
  - Trasformazioni delle informazioni in tutte quelle realtà che ne fanno uso
    - Non necessariamente devono essere eseguite dagli elaboratori
    - Il vantaggio dell'utilizzo degli elaboratori è dato dalla loro velocità e affidabilità

Linguaggi di programmazione

- Un po' di storia…
  - **Fortran**
    - Il primo linguaggio
      - Sviluppato a partire dal 1954, rilasciato nel 1957
      - Pensato per facilitare la scrittura di formule matematiche
  - **C**
    - Sviluppato nel 1972 da Dennis Ritchie presso i laboratori della AT&T Bell
      - Eredita principi e idee dal linguaggio B che a sua volta aveva ereditato da BCPL e CPL
      - Pensato per operare ad alto livello indipendentemente dalla macchina
      - Standard completato nel 1989 (ANSI C)
  - **C++**
    - È una estensione del C
    - Formulato da Bjarne Stroustrup all'inizio degli anni '80, sempre presso i laboratori della AT&T Bell
    - Supporta la programmazione orientata agli oggetti (OOP)

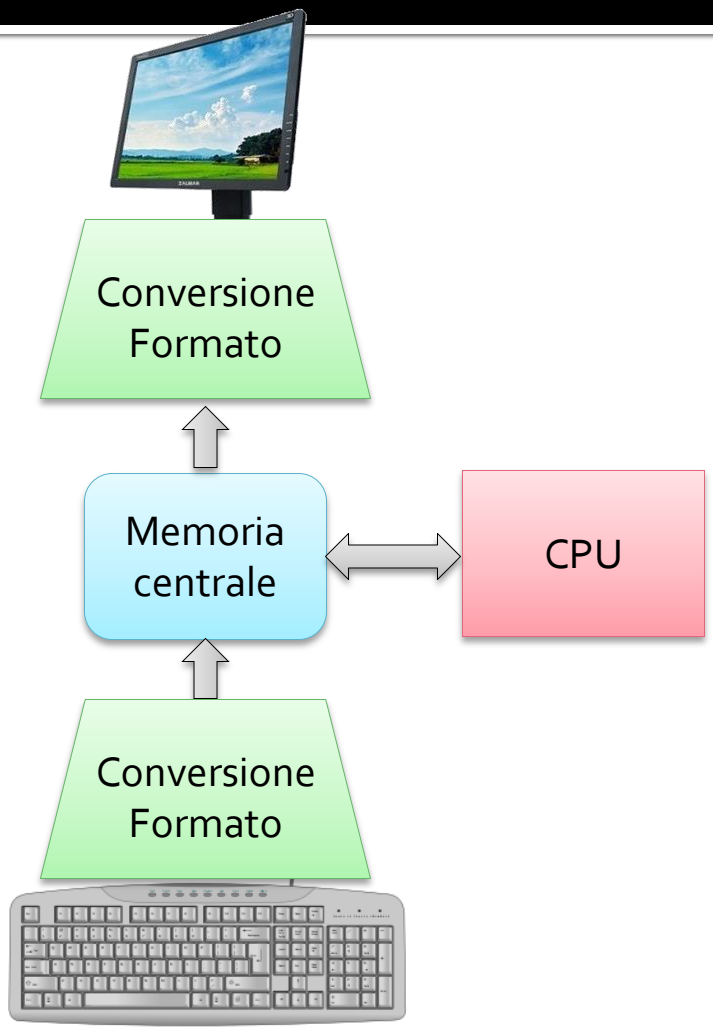
Il modello di esecutore

- Componenti fondamentali di un **computer**
  - Unità Centrale di Elaborazione (CPU)
    - Comprende ed esegue le istruzioni del programma
  - Memoria centrale
    - Contiene istruzioni e dati che servono all'esecuzione del programma
  - Dispositivi di input
    - Per inserire dati e istruzioni in memoria
  - Dispositivi di output
    - Per mostrare i risultati
  - Memoria di massa
    - Contiene i dati e le istruzioni che vengono inseriti nella memoria centrale prima e durante l'esecuzione del programma
    - Può conservare i risultati prodotti
    - Dispositivi sia di input che di output



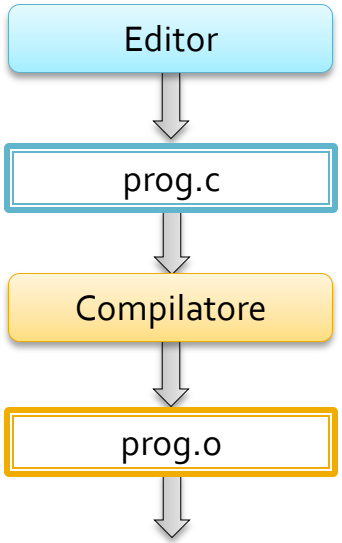
Il modello di esecutore

- **Input e output**
  - Stream di caratteri preso dalla codifica ASCII
    - in numero teoricamente infinito se
      - acquisito da tastiera (input standard)
      - restituito su terminale (output standard)
    - in numero finito se
      - scritto o letto in memorie di massa (file)
  - Conversione da stream di caratteri in binario e viceversa



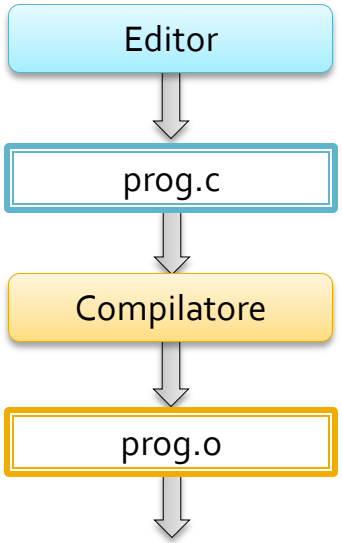
Compilazione dei programmi

- Per essere eseguito da una CPU, un algoritmo deve essere espresso in linguaggio macchina
  - Insieme di istruzioni che la CPU comprende
    - Sequenze di bit
- C e C++ sono linguaggi *di alto livello*
  - La loro potenza espressiva è superiore a quella del linguaggio macchina
  - Pensati per aiutare i programmatori
- Per tutti i linguaggi di programmazione esiste
  - La **grammatica**
  - Il programma di **traduzione** in linguaggio macchina
- Traduzione
  - **Sorgente**: linguaggio da cui parte la traduzione
  - **Oggetto**: linguaggio in cui si traduce



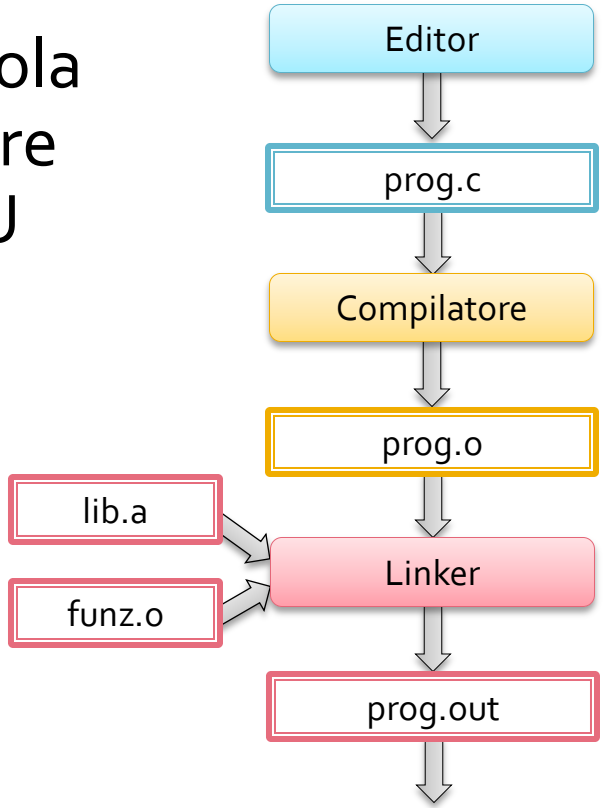
Traduzione dei programmi

- Compilazione
  - La traduzione avviene una volta sola
    - Velocità di esecuzione
    - Il programma dipende strettamente dalla CPU per la quale è stato prodotto
- Interpretazione
  - La traduzione avviene ogni volta che il programma viene eseguito
    - Tempi più lunghi
    - Maggiore portabilità



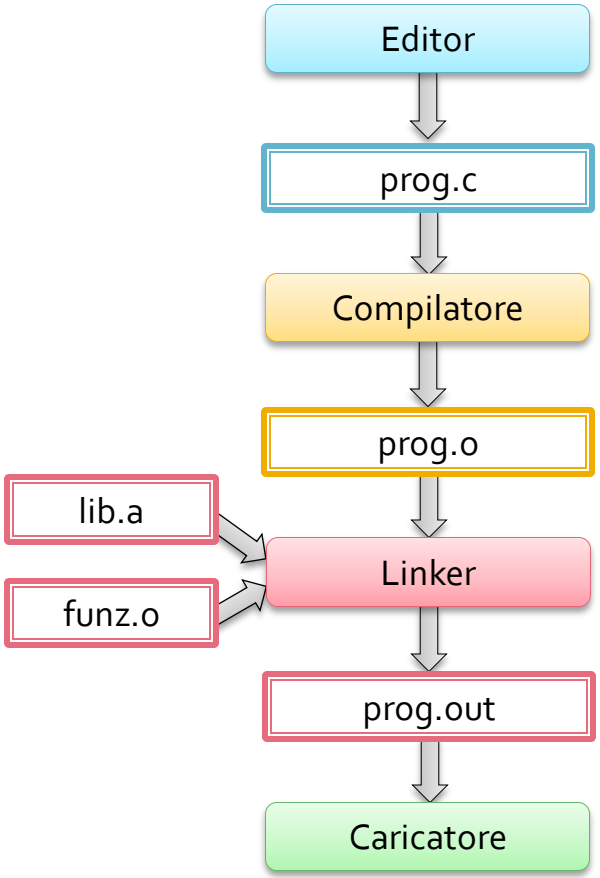
Compilazione dei programmi

- Nell'approccio per compilazione la sola traduzione non è sufficiente a rendere il programma «eseguibile» dalla CPU
  - Servono delle funzionalità raccolte in librerie
    - Interazione con il sistema operativo
    - Gestione dell'I/O
- Linker
  - Assembla tutti gli oggetti e librerie necessari per generare un programma che sia eseguibile dalla CPU



Compilazione dei programmi

- Loader o Caricatore
  - Carica in memoria il programma
  - Attiva il programma
- Il suffisso dei file dipende dal sistema operativo ospitante
  - *Windows*
    - Sorgente: .c / .cpp
    - Oggetto: .obj
    - Libreria: .lib
    - Eseguibile: .exe
  - *Unix*
    - Sorgente: .c / .cpp
    - Oggetto: .o
    - Libreria: .a
    - Eseguibile: .out



Compilazione dei programmi

- Compilazione a linea di comando Unix
  - Editore di testo
    - GNU Emacs
    - `$ emacs programma.c`
      - Crea (o apre se esistente) il file *programma.c* per editarlo
  - Compilatore
    - `$ gcc programma.c`
      - Produce il file eseguibile *a.out*
    - `$ gcc –o esegui programma.c`
      - Il file eseguibile viene nominato *esegui* e non *a.out*
    - gcc esegue la compilazione in tre passi
      - Un preprocessore modifica il sorgente
      - Il compilatore lo traduce in linguaggio macchina
      - Il linker crea l'eseguibile con il codice oggetto prodotto dal compilatore e quello disponibile nelle librerie
  - Esecuzione
    - `$ esegui`

Compilazione dei programmi

- Ambienti di sviluppo integrato (IDE)
  - In essi sono presenti funzionalità per
    - Scrivere e modificare codice sorgente (editor)
      - Gli editor sono sensibili alle parole chiave
        - Codice più facile da leggere
    - Tradurre il codice sorgente (compilatore)
    - Generare l'eseguibile (linker)
    - Effettuare il debugging
  - Utilizzeremo **Dev-C++**
    - IDE che funziona da front-end verso i compilatori *gcc* e *g++* pienamente compatibili con gli standard ANSI C e ANSI C++
    - Versione:
      - Orwell Dev-C++
    - Sito web:
      - http://sourceforge.net/projects/orwelldevcpp

Compilazione dei programmi

- Gestione degli errori
  - Classificazione
    - Errori che si verificano
      - Nella compilazione del sorgente
      - Nel collegamento di oggetti e librerie
      - Nel caricamento dell'eseguibile in memoria
      - Nell'esecuzione

      } Il programma non viene eseguito
  - Errori più frequenti
    - Nella compilazione
      - Assenza del ";" per chiudere le istruzioni
      - Mancanza di parentesi () per una funzione senza parametri
      - Errori dovuti al Case Sensitive
    - Nel collegamento
      - Uso di un elemento in un sorgente diverso dal modo in cui è definito in un altro sorgente
    - Nel caricamento
      - Un programma necessita di più memoria di quella esistente
    - Nell'esecuzione (anche chiamati eccezioni)
      - Tipicamente sono errori logici
        - Es.: divisione per 0, operazioni ripetute all'infinito (loop)

C e C++ e la progettazione di programmi

- Il legame tra la progettazione dei programmi ed il linguaggio è meno stretto di quanto si possa pensare
  - La potenza espressiva del linguaggio può essere di aiuto, ma è il fattore meno determinante per apprendere i fondamenti della programmazione
- Programmazione strutturata
  - Insieme di regole da seguire per progettare un programma di qualità e da adottare indipendentemente dal linguaggio di programmazione
- Qualità
  - Un compromesso tra obiettivi diversi:
    - Correttezza
    - Efficienza
    - Robustezza
    - Affidabilità
    - Usabilità
    - Estendibilità
    - Riusabilità
    - Strutturazione
    - Leggibilità
    - Manutenibilità
    - Modificabilità
    - Portabilità

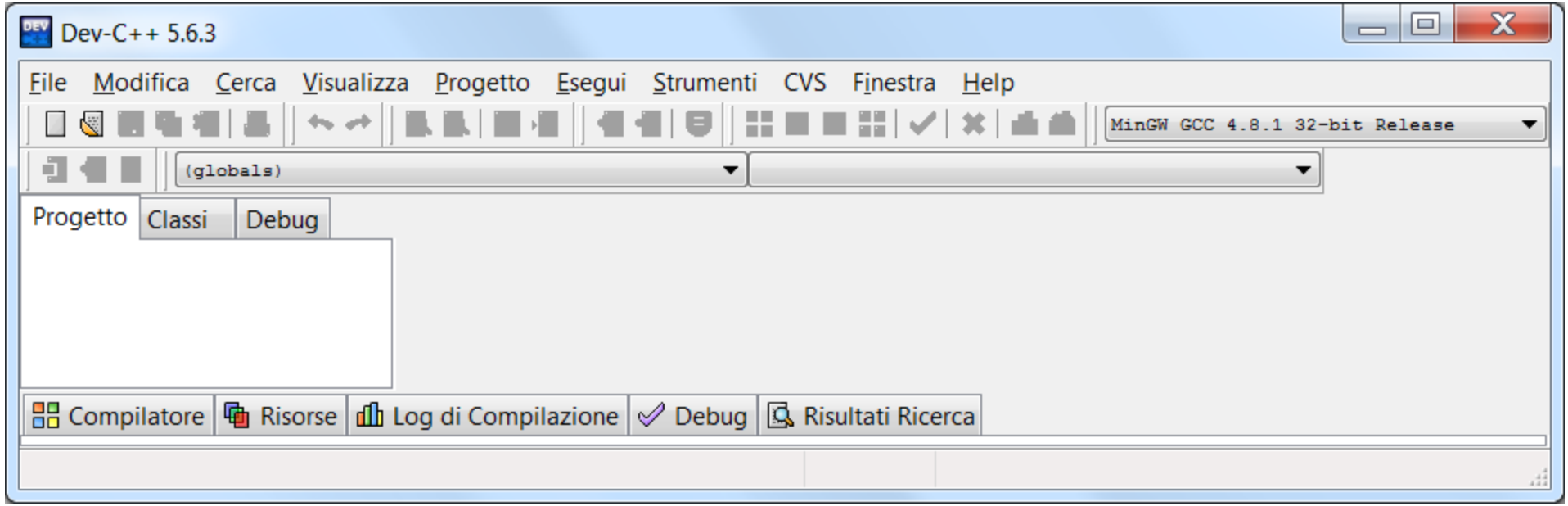
C e C++ e la progettazione di programmi

- La progettazione di programmi è un'attività complessa
  - Separazione netta tra
    - Cosa → Analisi dei requisiti e specifiche funzionali
    - Come → Progetto a diversi livelli di dettaglio
  - Principi fondamentali
    - Modularità
    - Uso di strutture di controllo one-in, one-out
    - Approccio top-down e stepwise refinement
      - Dal generale al particolare per raffinamenti successivi
      - Metodo deduttivo più adatto agli esseri umani
    - Approccio bottom-up
      - Da moduli elementari a moduli più complessi passando per integrazioni successive
      - Metodo induttivo

Iniziamo a programmare …per davvero

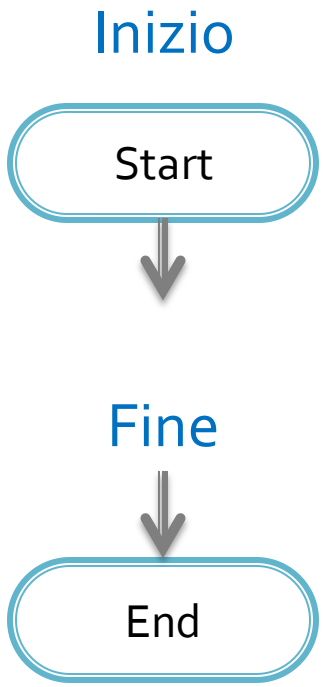
Iniziare a programmare

- **Ambiente IDE Dev-C++**
  - Installazione
  - Attivazione dell'ambiente da sistema operativo
  - Creazione di un nuovo programma sorgente
  - Salvare il programma su file dandogli un nome
  - Usare l'editor per scrivere/modificare il programma
  - Compilazione
  - Esecuzione

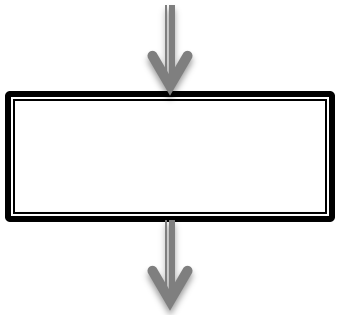


Diagrammi di flusso

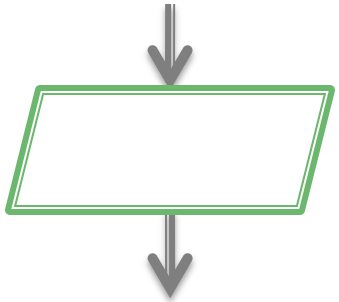
- I diagrammi di flusso o flow chart sono un formalismo che consente di rappresentare graficamente gli algoritmi
  - descrivono le azioni da eseguire ed il loro ordine di esecuzione
- Ogni azione corrisponde ad un simbolo grafico (blocco)
  - ogni blocco ha un ramo in ingresso ed uno o più rami in uscita



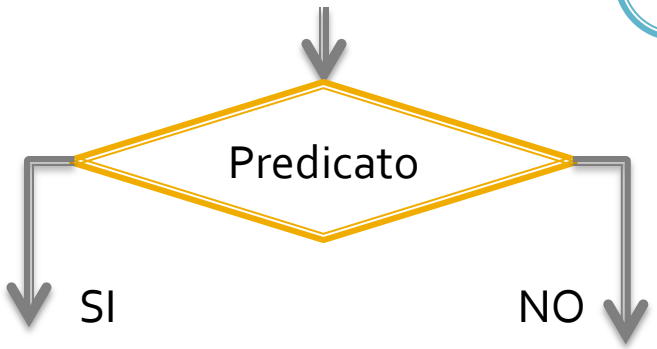
Elaborazione



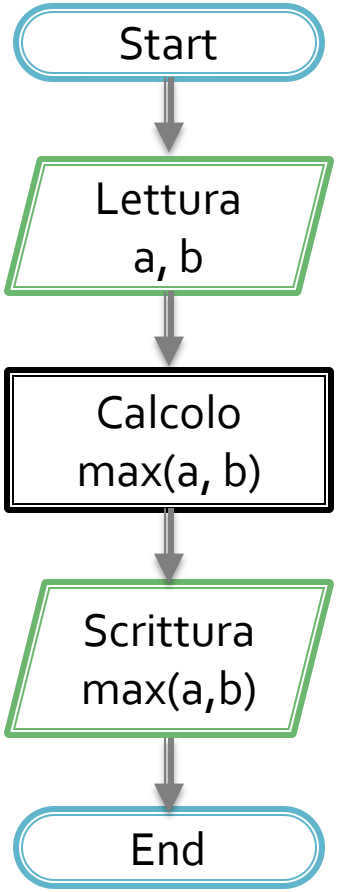
I/O



Selezione a due vie

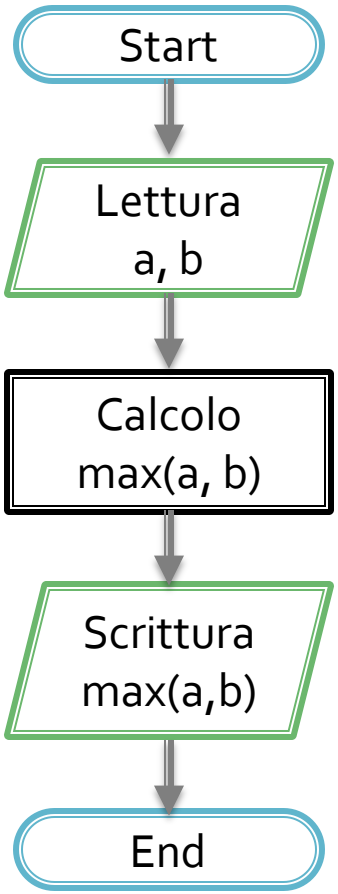


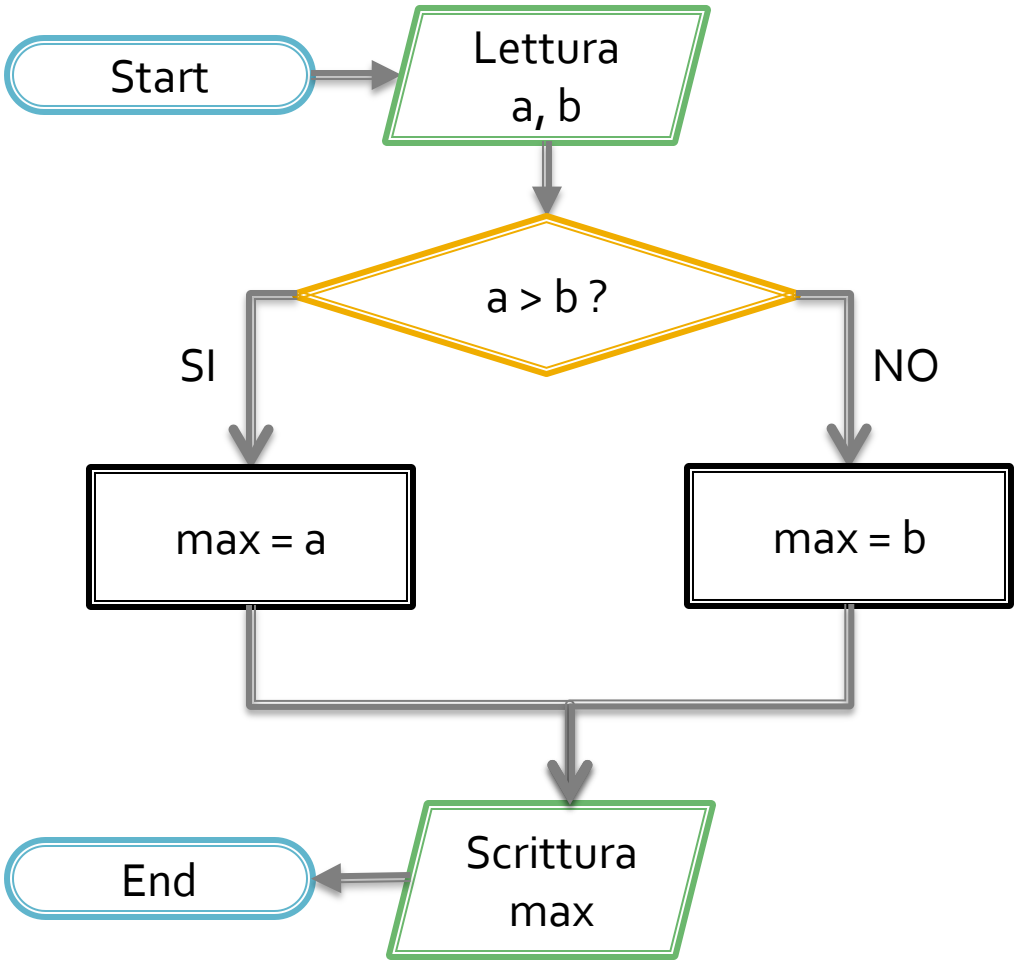
Esempio:
massimo tra due numeri



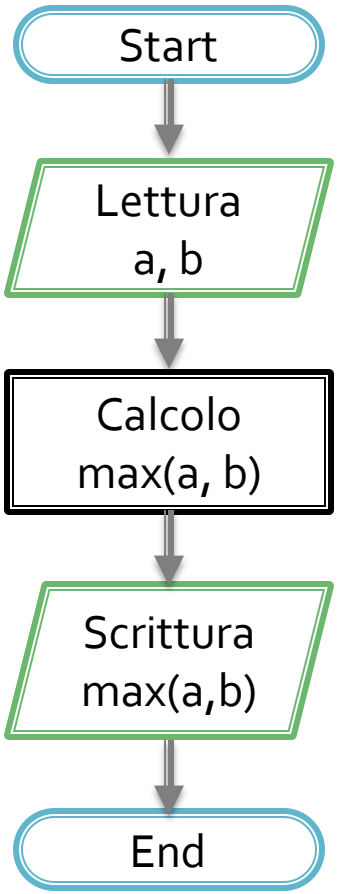
?

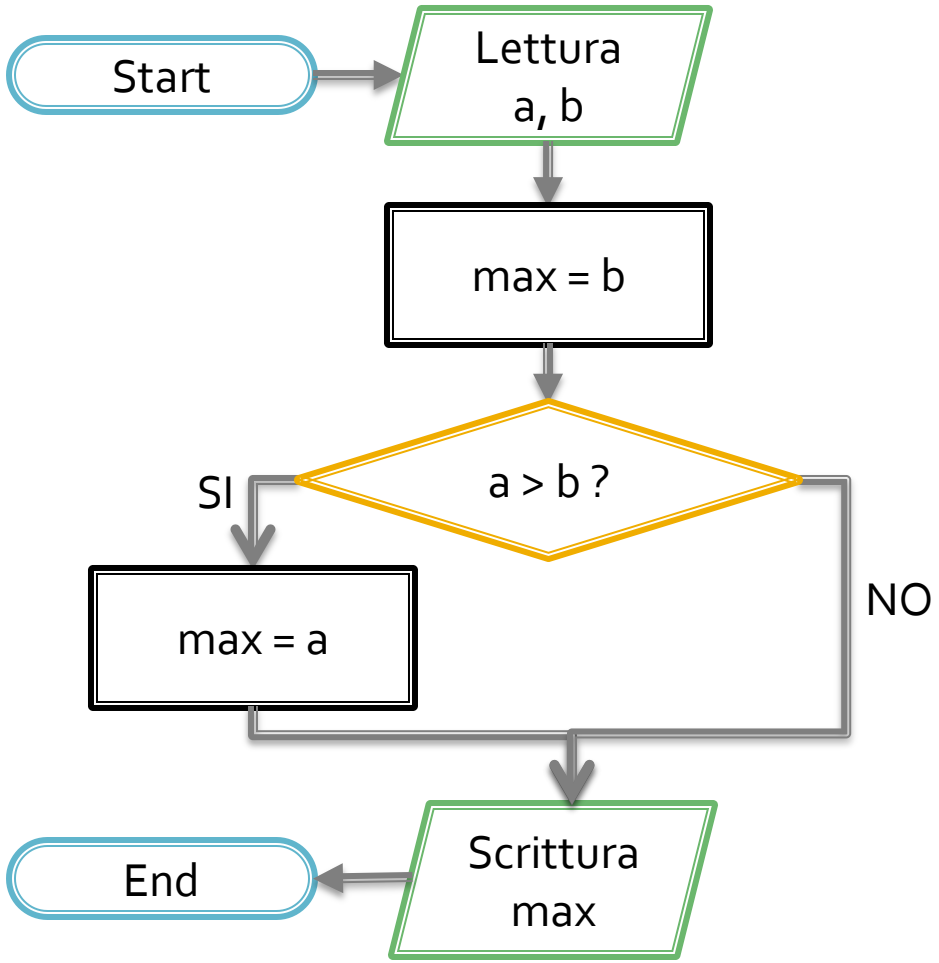
Esempio:
massimo tra due numeri





Esempio:
massimo tra due numeri (variante)





Esercizi

- Massimo tra tre numeri
  - Progettare l'algoritmo

Domande