Università degli Studi di Ferrara

Corso di Laurea in Chimica - A.A. 2018 - 2019

Programmazione
Lezione 3 – Introduzione alla Programmazione

Docente: Lorenzo Caruso - lorenzo.caruso@unife.it

Nelle lezioni precedenti

- Il software principale in un computer è il sistema operativo, e permette l'esecuzione dei programmi fornendo un accesso sicuro e controllato alle risorse
- Il kernel è la parte del S.O. che si occupa di gestire le risorse, è nota come nucleo del sistema
- La shell è l'interprete dei comandi in un sistema operativo, può essere grafica o testuale e sostanzialmente è la parte del sistema con cui l'utente comunemente interagisce
- Un sistema time sharing divide il tempo di esecuzione in quanti, assegnati ai diversi processi simulandone l'esecuzione contemporanea
- Lo scheduler è quella parte del S.O. che si occupa di assegnare la CPU ai diversi processi in esecuzione secondo una politica definita
- Il gestore della memoria è quella parte del S.O. che si occupa di assegnare la memoria virtuale ai processi in esecuzione, di gestire i cambi di contesto e di mappare la memoria virtuale sulla memoria fisica
- Il File System definisce gli strumenti necessari alla manipolazione, gestione e organizzazione di file in una memoria di archiviazione

In questa lezione

- Problemi e Calcolatori
- Algoritmi
- I Linguaggi di Programmazione
- Linguaggi interpretati e compilati

Il Computer

Riprendiamo la definizione: un computer è un dispositivo elettronico in grado di ricevere, trasmettere, immagazzinare ed elaborare informazioni ad alta velocità e con precisione.

Di conseguenza un computer è un dispositivo in grado di dare risposte rapide e corrette?

NO

Un computer è un dispositivo POTENZIALMENTE in grado di dare risposte rapide e corrette

Potenzialmente?

Affinché un computer possa garantire una risposta rapida e precisa ad un problema è necessario che:

- stia eseguendo un programma scritto da un bravo programmatore
- il programma sia eseguito da un bravo utente

Problemi e Calcolatori

- Abbiamo visto come i calcolatori siano stati sviluppati per svolgere calcoli al fine di risolvere problemi
- Abbiamo visto che il vantaggio nell'utilizzo di un calcolatore risiede nella sua capacità di eseguire operazioni elementari molto velocemente

  - Ogni calcolatore è in grado di comprendere ed eseguire un limitato e specifico insieme di istruzioni

Problemi e Calcolatori

Per poter risolvere un problema con un calcolatore è pertanto necessario:

- **Definire** i passaggi che risolvono il problema in una forma comprensibile all'essere umano
- **Ridefinire** questi passaggi in modo che essi possano essere compresi dal calcolatore

Algoritmi

- Il primo passo è quello di definire la strategia risolutiva del problema per **passi successivi**
- L'insieme dei passi successivi in cui viene ridefinito il modo di risolvere il problema, prende il nome di

ALGORITMO

Algoritmi

Originariamente il termine è stato coniato per descrivere il procedimento per il calcolo delle operazioni dell'aritmetica elementare.
In seguito il termine venne utilizzato per identificare qualsiasi procedimento di calcolo.

- Un Algoritmo esprime le azioni (istruzioni) da svolgere su determinati oggetti (costanti o variabili) al fine di produrre gli effetti desiderati.

Algoritmi

Un Algoritmo, pertanto, è una funzione di trasformazione dei dati:

A (input) => B (output)

Algoritmi

Problema

↓

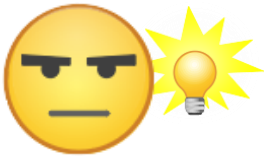
Ridefinizione

↓

Algoritmo

Area di un rettangolo

↓



↓

- Assegna a **base** il valore 3
- Assegna ad **altezza** il valore 7
- Assegna ad **area** il valore **base***altezza**
- Visualizza **area**

Proprietà di un Algoritmo

- Il numero di istruzioni che fanno parte di un algoritmo è finito.
- Le istruzioni presenti in un algoritmo devono essere definite senza ambiguità: un algoritmo eseguito più volte e da diversi esecutori, a parità di premesse, deve giungere ai medesimi risultati (principio di determinismo).
- Un algoritmo si deve occupare della risoluzione di famiglie di problemi (principio di generalità).
- Tutte le azioni descritte devono essere eseguibili con i mezzi di cui si dispone.

Rappresentazione di Algoritmi

Possiamo descrivere gli algoritmi in tre modi:

- Descrizione di alto livello: Descrivere un algoritmo ignorando i dettagli dell'implementazione. Non dobbiamo preoccuparci di come funzionano le macchine.
- Implementazione: Definisce come farebbe la macchina ad eseguire l'algoritmo, come leggere i dati e calcolare il risultato.
- Descrizione formale: E' la descrizione più dettagliata e vicina alla macchina.

Astrazione

Nella rappresentazione di un algoritmo utilizzare un alto livello di astrazione significa scegliere una descrizione intuitiva per le persone, questo offre alcuni vantaggi e svantaggi:

Pro:
- Intuitivo per le persone
- Ovvia correttezza dell'algoritmo
- Utile per schematizzare

Contro:
- Non si affrontano problemi tecnici
- In generale non piace (si tendono a mascherare eccessivamente possibili problemi implementativi)
- Se è troppo astratto potrebbero esserci complicazioni

Astrazione

Scegliere un basso di livello di astrazione porta a problematiche invertite:

- vengono sacrificate intuitività e facilità di realizzazione per favorire la successiva traduzione in programma

Algoritmi: elementi base

Possiamo pensare agli elementi base che consentono la rappresentazione di algoritmi come ai mattoncini di un set di costruzione Lego:

Ogni mattoncino ha una propria funzione specifica (muro, porta, finestra) e vengono composti per ottenere la costruzione desiderata

Algoritmi: elementi base

Un elemento fondamentale sono **le variabili**

Gli algoritmi operano principalmente su variabili che conterranno i dati sui quali si vuole svolgere una determinata elaborazione.

I valori da elaborare devono essere assegnati alle variabili prima di effettuare l'elaborazione.

Le istruzioni operano sui valori contenuti: se questi non ci sono non ci si può attendere alcuna elaborazione.

Ogni variabile è identificata da un nome che permette di distinguerla dalle altre, indipendentemente dal valore contenuto.

Algoritmi: elementi base

Tipi di istruzioni

- Input (da console, file, rete, ...)
  - es: “leggi m”
- Output (su console, file, rete, ...)
  - es: “visualizza m”
- Assegnamento (modifica del valore)
  - es: “m = 3”

Algoritmi: controllo del flusso

La **sequenza**: specifica l'ordine in cui le istruzioni si susseguono

La **selezione**: permette di scegliere fra due alternative la sequenza di esecuzione

L'**iterazione**: permette di ripetere più volte la stessa sequenza fino al verificarsi di una condizione

Algoritmi: controllo del flusso

Le strutture di controllo devono essere pensate come schemi di composizione:

- Una sequenza può contenere iterazioni e selezioni
- Una iterazione può contenere a sua volta sequenze e selezioni
- Una selezione può contenere a sua volta sequenze e iterazioni

➔ Le strutture di controllo sono i componenti fondamentali per costruire algoritmi di varia complessità

Sequenza

Dall’esempio visto prima

- Leggi il valore di Base
- Leggi il valore Altezza
- Assegna ad Area il valore Base * Altezza
- Scrivi Area

Selezione

Leggi il valore di *Dividendo*
Leggi il valore *Divisore*
Se *Divisore* è maggiore di 0 **allora**
Assegna a *Quoziente* il valore *Dividendo* / *Divisore*
Scrivi *Quoziente*
Altrimenti
Scrivi “Non posso eseguire la divisione”
Fine

Iterazione

Supponiamo (per esempio) di voler calcolare il quadrato di una serie di numeri positivi

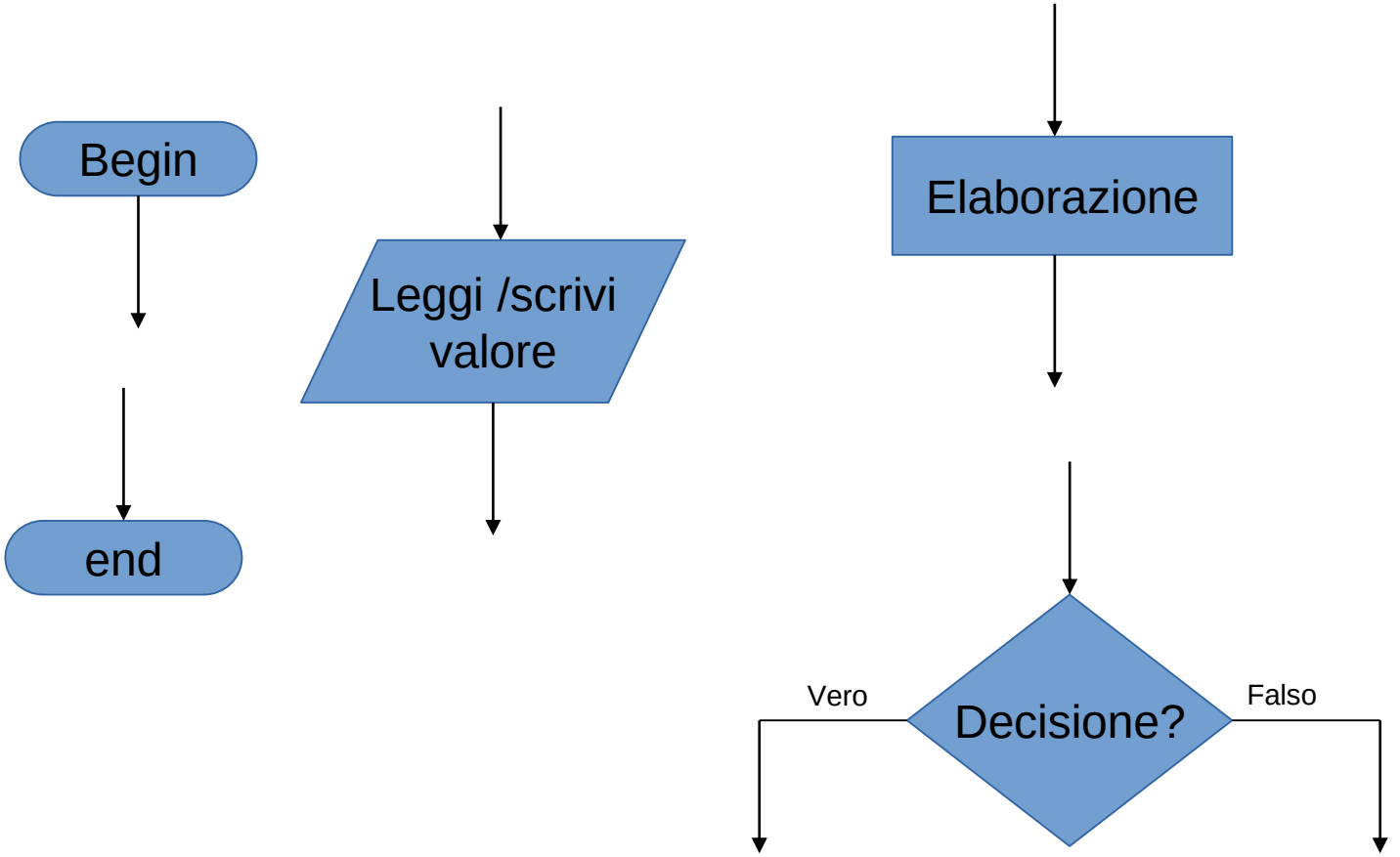
Leggi il valore di Numero
Finché Numero > 0
Assegna a Quadrato il valore Numero * Numero
Scrivi Quadrato
Leggi il valore di Numero
Fine

Programmazione strutturata

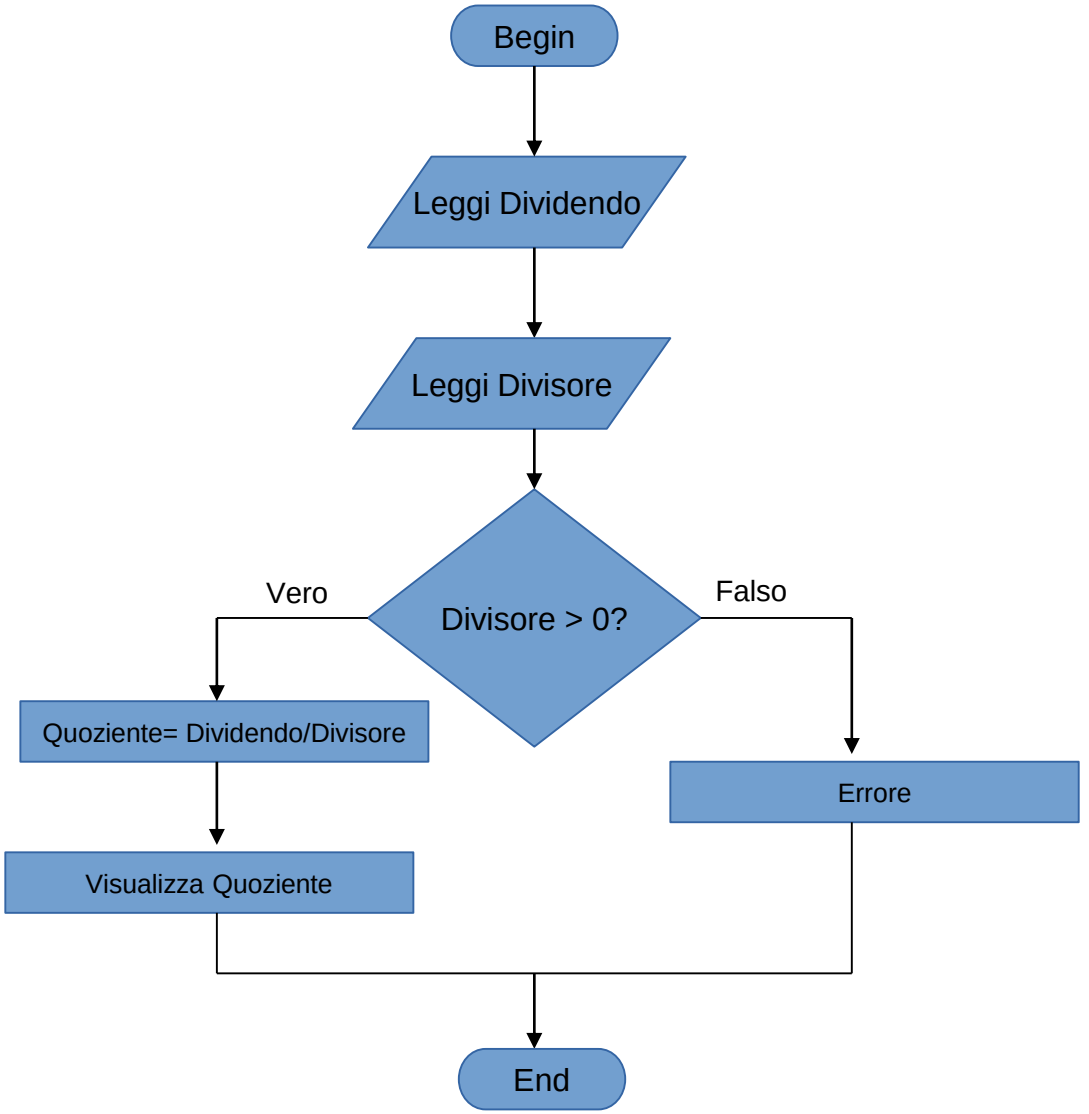
L’utilizzo di questi elementi nella rappresentazione di un algoritmo è detto **programmazione strutturata**

la Programmazione strutturata fornisce, a partire dagli anni 60, le regole per la scrittura di “buoni” algoritmi, ovvero le indicazioni appena viste.

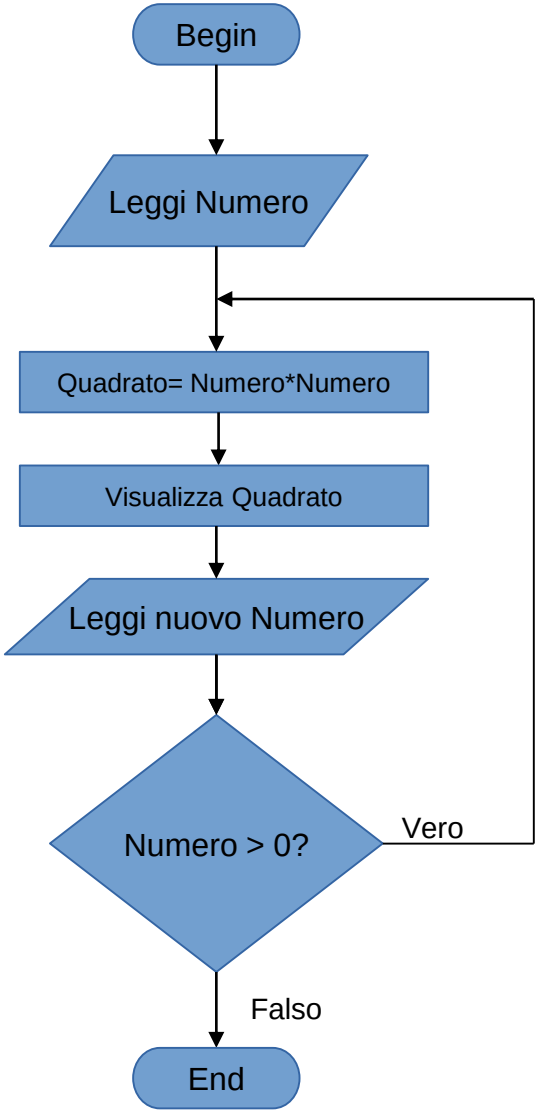
Flow chart


Begin
end
Leggi /scrivi valore
Elaborazione
Vero
Decisione?
Falso

Flow chart: divisione



Flow chart: Quadrato



Accumulatori

Gli accumulatori, nel linguaggio della programmazione sono variabili nelle quali il nuovo valore non sostituisce il valore precedente, ma si somma (accumula) a quello già presente.

Se all'accumulatore viene "passata" la seguente sequenza:

1, 2, 3, 4

Allora l'accumulatore assumerà i seguenti valori***** :

1, 3, 6, 10

Se il valore di incremento dell'accumulatore resta costante di una unità, allora otteniamo un **Contatore**

***** Supponendo che il contatore sia inizializzato a 0

Dal Problema al Programma

A questo punto il nostro Algoritmo è espresso in un modo più vicino a quello che un calcolatore è in grado di eseguire, ma non è “scritto” in una lingua che il calcolatore è in grado di comprendere.

Per essere eseguito da un calcolatore, il nostro Algoritmo deve essere codificato con un **linguaggio di programmazione**.

Il nostro Algoritmo codificato prende il nome di:

Codice Sorgente

Dal problema al programma

Una volta codificato, il **Codice Sorgente**, deve subire una ulteriore trasformazione prima di poter essere eseguito, deve essere tradotto in **Codice Macchina** (unico linguaggio che la macchina è realmente in grado di comprendere).
La traduzione dal codice sorgente al codice macchina, è un processo che viene eseguito da strumenti specifici (compilatori ed interpreti) ed ogni linguaggio di programmazione ha il proprio strumento.

Il Linguaggio Macchina

Il linguaggio macchina costituito da zero ed uno è l'unico che comanda direttamente le unità fisiche dell'elaboratore in quanto è l’unico comprensibile dall’elaboratore stesso.

È però estremamente complicato scrivere programmi in tale linguaggio.

Per poter permettere un dialogo più semplice con la macchina sono nati i linguaggi di programmazione.

Linguaggi di Programmazione

Un linguaggio è composto da:

- un insieme di parole raggruppate in un dizionario
- un insieme di operatori per unire tra loro le parole
- un insieme di regole sintattiche - regole che permettono di unire insieme le parole per ottenere delle frasi corrette
- un insieme di regole semantiche - regole che permettono di dare un senso alle frasi del linguaggio

Linguaggi di Programmazione

- Un algoritmo scritto in un linguaggio di programmazione viene chiamato **Programma**
- Il processo di scrittura del programma, a partire dall'algoritmo, viene chiamato **Codifica**

Linguaggi Compilati e Interpretati

Abbiamo visto come un codice sorgente necessiti di uno strumento di traduzione che trasformi il codice in linguaggio macchina.
Su questi strumenti esiste una importante distinzione:

Linguaggi Compilati → uno strumento chiamato compilatore traduce in linguaggio macchina il codice sorgente. Questo avviene in maniera indipendente dall'esecuzione.
• Pro: velocità di esecuzione
• Contro: eseguibile vincolato all'architettura sulla quale è stato compilato

Linguaggi Interpretati → durante l'esecuzione del programma, le istruzioni vengono analizzate ed eseguite riga per riga da uno strumento chiamato interprete.
• Pro: portabilità
• Contro: lentezza di esecuzione

Linguaggi Compilati e Interpretati

I linguaggi compilati necessitano di un programma (il compilatore) che analizzerà il codice sorgente traducendolo in un linguaggio comprensibile solo al calcolatore **producendo un nuovo file:**

- questo file viene chiamato file binario o eseguibile contiene il nostro algoritmo tradotto sotto forma di istruzioni elementari che il calcolatore è in grado di eseguire

I linguaggi interpretati, per poter essere eseguiti, necessitano di un programma (l'interprete) che tradurrà le righe di codice e le eseguirà sulla macchina all'istante

- l'interprete **non produce un nuovo file**, ma legge, interpreta ed esegue le linee di codice sorgente come unica operazione

Linguaggi di basso livello

Il più vecchio linguaggio di programmazione è il linguaggio **assembly**.
Il linguaggio assembly è una rappresentazione simbolica del linguaggio macchina. La scrittura di programmi è enormemente semplificata rispetto a quest'ultimo.

Per essere eseguito dall'elaboratore un programma in linguaggio assembly deve essere tradotto in linguaggio macchina; tale lavoro è a carico di un programma detto assemblatore.

Questi due tipi di linguaggi, detti anche linguaggi di basso livello sono propri di ogni macchina, cioè ogni tipo di processore ha un proprio linguaggio assembly.

Linguaggi di alto livello

I linguaggi di alto livello sono più vicini al linguaggio naturale, sono orientati ai problemi piuttosto che all'architettura della macchina.
Non fanno riferimento ai registri fisicamente presenti sulla macchina ma a variabili.
Per essere eseguiti devono essere tradotti in linguaggio macchina, e tale traduzione viene svolta da un programma detto compilatore.
I linguaggi di alto livello sono in larga misura indipendenti dalla macchina, possono essere eseguiti su qualsiasi elaboratore a patto che esista il corrispondente compilatore che ne permetta la traduzione.

Caratteristiche di un Linguaggio

- Tipizzazione statica/dinamica
- Gestione automatica /manuale della memoria (garbage collection)
- Immutabilità
- Impostazione procedurale / dichiarativa (funzionale, logica)
- Organizzazione del codice strutturata / orientata agli oggetti

Grazie per l’attenzione

Riferimenti

Il corso di programmazione per il primo anno della Laurea Triennale in Matematica nasce con l'intento di unire ai principi di programmazione una conoscenza basilare di uno degli strumenti software più diffusi nell'ambito matematico: Matlab.
La prima parte del corso pertanto è una rielaborazione del programma di Programmazione per la Laurea Triennale in Informatica 15/16 (in particolare) e 16/17.
Parte del materiale originale da cui ho ricavato il percorso didattico, alcuni approfondimenti ed integrazioni possono essere trovati in:

- Lezioni di Programmazione 15/16 CdS Informatica - Giacomo Piva
- Lezioni di Programmazione 16/17 CdS Informatica - Marco Alberti