

UNIVERSITÀ DI GENOVA
FACOLTÀ DI SCIENZE M.F.N.
Dispense del corso di Laboratorio di Calcolo A
a.a. 2001-2002

Patrizia Boccacci, Paolo Morettini
Claudia Gemme, Maurizio Lo Vetere, Fabrizio Parodi

21st September 2001

HIGZ – High level Interface to Graphics and Zebra

HPLOT – User's Guide

HBOOK – Statistical Analysis and Histogramming

MINUIT – Function Minimization and Error Analysis

CERN Program Library entry **Y250 Y251 D506**

Copyright CERN, Geneva 1991

Copyright and any other appropriate legal protection of these computer programs and associated documentation reserved in all countries of the world.

These programs or documentation may not be reproduced by any method without prior written consent of the Director-General of CERN or his delegate.

Permission for the usage of any programs described herein is granted a priori to those scientific institutes associated with the CERN experimental program or with whom CERN has concluded a scientific collaboration agreement.

Requests for information should be addressed to:

CERN Program Library Office
CERN-CN Division
CH-1211 Geneva 23
Switzerland
Tel. +41 22 767 4951
Fax. +41 22 767 8630
Email: cernlib@cern.ch

Trademark notice: All trademarks appearing in this guide are acknowledged as such.

Table of Contents

1	Introduzione	6
1.1	Che cos' è un programma	6
1.2	Che cos' è un algoritmo	6
2	Configurazione base di un calcolatore	9
2.1	Come è fatto un PC	9
2.2	Scheda Madre	11
2.2.1	Il clock di sistema	13
2.2.2	Unitá a dischetti floppy	13
2.2.3	Spie luminose e dispositivi assortiti	14
2.2.4	Batteria tampone	14
2.2.5	Porte seriali	14
2.2.6	Porta parallela	14
2.2.7	Universal Serial Bus (USB)	14
2.2.8	Connettori al bus IDE	15
2.3	CPU	15
2.4	La memoria centrale	16
2.5	Hard disk	17
3	Il sistema operativo	19
3.1	Livelli di funzionalità di un calcolatore	19
3.2	Linux	19
3.2.1	Un po' di storia	19
3.2.2	La struttura di Linux: il kernel e la shell	21
3.3	Struttura dei files e delle directories	21
3.4	Controllo dei processi	23
3.5	Comandi per la gestione di files e directories	23
3.5.1	Utilizzo di wildcard	23
3.5.2	Pipe e redirectione	25
3.5.3	Protezioni	25
3.6	Il text editor	26
3.7	Compilazione ed esecuzione di un programma	26
3.7.1	Il comando make	27
3.8	Il debugger simbolico	27
3.9	Procedure di comandi	28
3.9.1	Uso delle variabili	29
3.9.2	Realizzazione ed esecuzione di una procedura di comandi	29
3.10	Archiviazione di files	30
3.10.1	Compressione di files	30
3.10.2	Accesso a dischi rimovibili	31
3.11	Come ottenere ulteriori informazioni	31
3.12	Il sistema grafico X11	31

Capitolo 1: Introduzione

1.1 Che cos' è un programma

Il termine programma può essere utilizzato per riferirsi a sequenze di azioni che si prefiggono il raggiungimento di certi obiettivi precisi: in un corso di studi il termine programma indica una sequenza temporale di argomenti da svolgere e di obiettivi didattici da raggiungere o in politica con termine programma di un partito si indicano una serie di azioni e di impegni da mantenere nell'immediato futuro.

Nonostante i numerosissimi casi in cui si utilizza il termine programma, uno degli usi più frequenti è diventato quello riferito al mondo dei calcolatori.

DEF. Per **programma** si intende *una sequenza di istruzioni scritte in un linguaggio comprensibile al calcolatore* che le esegue per ottenere i risultati richiesti.

L'esecutore del programma è quindi il calcolatore. Nel 1642 il francese Blaise Pascal inventa una macchina da calcolo (chiamata pascalina) per aiutare il padre nel suo lavoro di funzionario delle imposte; il principio di funzionamento si basava sul movimento di ruote dentate.

All'inizio del Novecento le macchine calcolatrici divennero da meccaniche ad elettriche. Nel 1952 l'ungherese John von Neumann realizza il primo vero prototipo dei moderni elaboratori elettronici che poteva essere programmato per risolvere problemi diversi.

L'elaboratore entra nel mondo commerciale e da allora con una crescita esponenziale aumentano le prestazioni e si riducono le dimensioni e i costi di produzione.

Nel 1965 Gordon Moore (uno dei fondatori della Intel) formulò una legge che sosteneva che ogni 18 mesi sarebbe raddoppiato il numero dei transistor contenuti nei circuiti integrati. Questa legge, che è rimasta valida anche nel nuovo millennio con una piccola correzione: da 18 mesi si è passati ad un anno. (informazioni ulteriori <http://www.intel.com/research/silicon/mooreslaw.htm>)

Potete notare che il grafico inizia nel 1970, due anni dopo lo sbarco (presunto) dell'uomo sulla Luna.

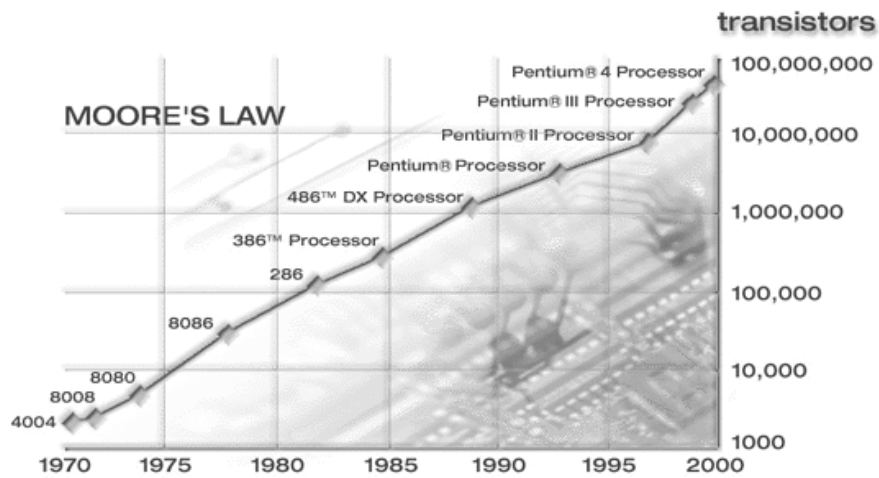
1.2 Che cos' è un algoritmo

Abbiamo appena accennato al concetto di programma nel campo degli elaboratori elettronici. Abbiamo detto che l'esecutore del programma è l'elaboratore elettronico e che il programma è una sequenza di istruzioni scritte in un linguaggio comprensibile al calcolatore. La *sequenza di istruzioni*, prima di essere tradotta in programma, rappresenta la *strategia* per raggiungere il risultato richiesto.

DEF. Si definisce **algoritmo** (dal nome del matematico arabo Al-Khuwarizmi) una serie finita di operazioni univocamente interpretabili, eseguite in sequenza, che trasforma i dati iniziali nel risultato richiesto.

Se il programma è la traduzione dell'algoritmo per poter essere eseguito senza errori dall'elaboratore questo deve essere univocamente interpretabile e deve essere rappresentato in una forma che ne consenta una facile conversione in programma.

L'insieme dei valori permessi dai dati in ingresso si definisce **dominio** dell'algoritmo (analogamente alle funzioni matematiche) mentre l'insieme dei valori che possono assumere i dati in uscita rappresenta il **codominio**.



In generale, ai fini di una migliore utilizzazione delle risorse disponibili (in termini di spazio e tempo) e per poter trattare problemi, anche di grandi dimensioni, con costi contenuti, è comunque fondamentale una scelta attenta nella formulazione dell'algoritmo risolutivo di un dato problema.

Una volta formulato l'algoritmo occorre verificare che questo fornisca:

- una soluzione per qualunque valore dei dati iniziali purchè appartenenti al dominio
- e che questa venga raggiunta in un numero finito di passi ovvero in un tempo finito.

E' chiaro che per descrivere un algoritmo funzionante devono essere considerati tutti i casi particolari anche con l'analisi dei risultati intermedi.

Una forma molto usata per la descrizione degli algoritmi è quella del *flow-chart*.

Esempio di algoritmo

Supponiamo di voler risolvere una equazione di secondo grado espressa in forma normale:

$$ax^2 + bx + c = 0 \quad (1.1)$$

Descriviamo questo semplice algoritmo prima a parole poi con un flow-chart.

1. Acquisisco a, b, c (supponiamo per semplicità che a sia diversa da 0)
2. calcolo il $\Delta = b^2 - 4ac$
3. Verifico se $\Delta < 0$?
4. Se la risposta alla domanda precedente è *vero*, stampo che non ci sono soluzioni reali e vado al passo 10; se la risposta è *falso* passo al punto successivo.
5. Verifico se $\Delta = 0$?
6. se la risposta alla domanda precedente è *vero*, pongo $x_1 = x_2 = -b/2a$, vado al passo 9; se la risposta è *falso* passo al punto successivo.
7. $x_1 = \frac{-b - \sqrt{\Delta}}{2a}$

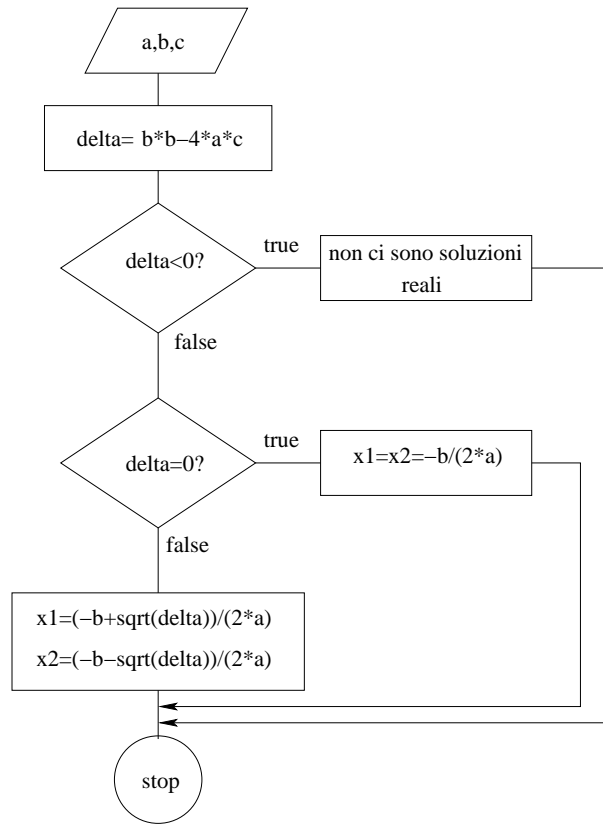


Figura 1.1: Grafico di flusso che descrive l’algoritmo per la risoluzione delle equazioni di secondo grado nel campo reale.

8. $x_2 = \frac{-b + \sqrt{\Delta}}{2a}$
9. stampo il risultato
10. termine della procedura