centro appunti
www.centroappunti.it
Corso Luigi Einaudi, 55 - Torino

Appunti universitari
Tesi di laurea
Cartoleria e cancelleria
Stampa file e fotocopie
Print on demand
Rilegature

NUMERO: 816 DATA: 04/02/2014

A P P U N T I

STUDENTE: Zorzi

MATERIA: Informatica

Prof. Poncino

Il presente lavoro nasce dall'impegno dell'autore ed è distribuito in accordo con il Centro Appunti.
Tutti i diritti sono riservati. È vietata qualsiasi riproduzione, copia totale o parziale, dei contenuti inseriti nel presente volume, ivi inclusa la memorizzazione, rielaborazione, diffusione o distribuzione dei contenuti stessi mediante qualunque supporto magnetico o cartaceo, piattaforma tecnologica o rete telematica, senza previa autorizzazione scritta dell'autore.

ATTENZIONE: QUESTI APPUNTI SONO FATTI DA STUDENTI E NON SONO STATI VISIONATI DAL DOCENTE.
IL NOME DEL PROFESSORE, SERVE SOLO PER IDENTIFICARE IL CORSO.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 3 di 100

INFORMATICA

prof. MASSIMO PONCINO : massimo.poncino@polito.it

esercit ELIAS CAROTTI : elias.carotti@polito.it

PROGRAMMARE → scrivere sul pc qualcosa che ha un c.d. x risolvere un problema

4.sh/settimana + 1.sh LAB (venerdì 11:30-13:00 LAIB 5)

ESAME

scritto → - 3 domande sulla teoria
- scrittura di un programma dic

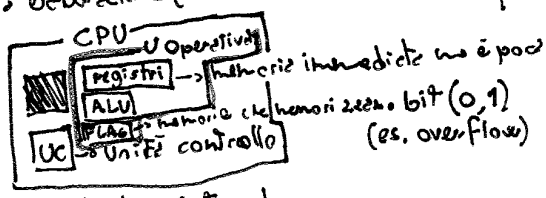
devono essere entrambe superate

PARTE TEORIA

3 domande → 6 punti

2/3 per sufficienza

- Niente storia
- sapere TRADUZIONE DI UN PROGRAMMA (come funziona la costruzione di un programma) (es. passi princ. della compilazione di un prog.?)
- modulo console: mentre in codeblocks scrittura, compilazione e esecuzione si fanno in automatico, in modo console si devono fare separate.
- i blocchi fondamentali dell'elaboratori: ingresso → unità elaborazione (che memorizza sulla memoria) → uscita
- CPU (central processing unit): schema fatto in classe →
- BUS (che cos'è un BUS)



- ALU (Aritmetic logic Unit): le operazioni: + - / * % & && || ! int e char
- FPU (Floating Point Unit): quando faccio calcoli reali. Unità di calcolo per numeri reali tipo float double
  + - / *
- CLOCK: segnale periodico distribuito su tutto il processore. Tutti gli strumenti del calcolatore si basano sui clock. Frequenza = O (GHz) → microprocessori dei PC etc (3/4 G) ma dipende dal dominio applicativo (dove lo utilizzo)
- TEMPISTICA DELLE ISTRUZIONI (p.103): i cicli macchina sono i cicli di clock e un'operazione richiede un numero intero di clock
- Unità di Controllo: è controlla le altre unità controllando l'esecuzione delle applicazioni (schema p. 119) Il programma risiede in memoria e la UC, attivata quando viene attivato il programma, opera tramite il PC (Program Counter → dice l'istruzione dove prelevare. Contiene l'indirizzo della prossima istruzione da eseguire), e l'IR (Instruction Register. Il PC parcheggia ciò che ha preso nell'IR.) e la logica di controllo (che attiva l'UO adeguata).

3 FASI: • FETCH IR ← M[PC]
PC ← PC+1
• Decode ordini ← decode(IR)
• Execute ready? Go!

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 3 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 5 di 100

c2 $\leftrightarrow$ ( )$_{10}$ $\quad -\underset{\substack{5\,4\,3\,2\,1\,0}}{1\,0\,1\,0\,1\,1} \rightarrow -1\cdot 2^5+0+1\cdot 2^3+0+1\cdot 2+1\cdot 2^0 = -32+0+2+1 = (-21)_{10}$

4) N bit devo sapere l'intervallo di rappresentazione: quanti n° posso rappresentare? 2^N valori:

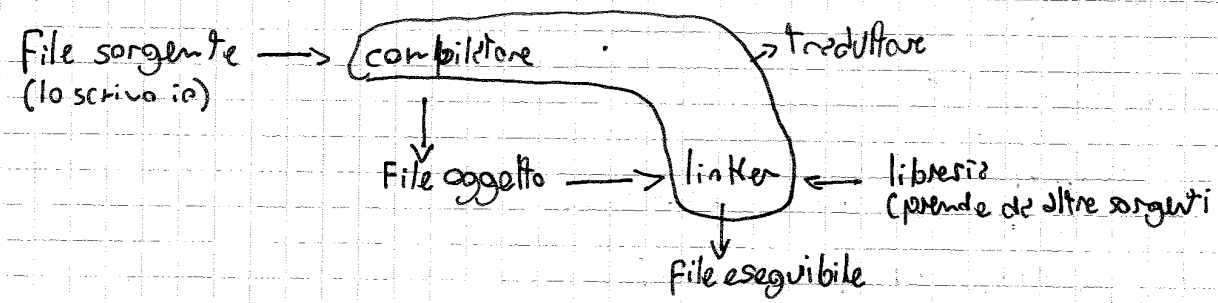
- bin. puro $\rightarrow$ $0 \dots 2^N-1$ (es. 6 $\rightarrow$ 0, 1, …, 63)
- m & s $\rightarrow$ $-2^{N-1}-1, \dots, -0, +0, \dots, 2^{N-1}-1$ es(-31, +…, +31)
- c2 $\rightarrow$ $-2^{N-1}, \dots, 2^{N-1}-1$ es(-32, …, +31)

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 5 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 7 di 100

SVILUPPO DI UN PROGRAMMA

- scrittura di un programma con un certo linguaggio
- traduzione di un programma in formato leggibile dal calcolatore attraverso un TRADUTTORE



PROGRAMMARE := scrittura di un file sorgente che descrive la soluzione del problema in oggetto.

Non esiste "la" soluzione ad un problema, non c'è un solo metodo standard o una sola soluzione.

È un operazione creativa ~~ma~~ complessa.

N.B
→ file

PROBLEMA (massimo comune divisore) → esperienza → SOLUZIONE → strumenti aut. e metodi formali → SOL. FORMALE ⇒ tutti comprensibile dal

SOLUZIONE → espressa a parole

SOL. FORMALE → tecnica di programmazione automatica

PROGRAMMA

.c → file sorgente

TRADUZIONE

.exe → file eseguibile

Si fa prima una soluzione informale che, attraverso un ALGORITMO viene resa comprensibile al calcolatore.

ALGORITMO := descrizione formale di una sequenza finita
... (slides)

PROBLEMA	calcolo il massimo tra A e B
SOLUZIONE	Il massimo è il + grande tra A e B
SOL. FORMALE	I Acquisisci A " B II se A>B max ← A (max=A) III se B>A max ← B (max=B)

⇓

PSEUDO-CODICE (quasi-programma)

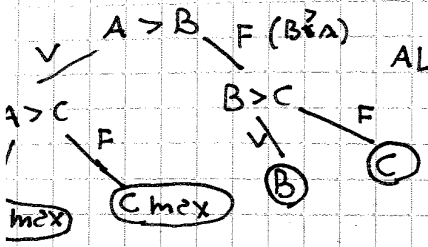
© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 7 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 9 di 100

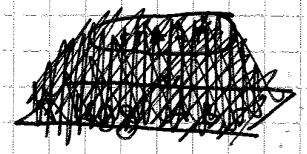
es) Problema: Dati 3 valori acquisiti in Input A,B,C. Calcola il massimo tra i 3

Sol. informale: confronta i valori in coppia ed in sequenza

N.B. VARIABILI (A,B,C) come blocchi



ALBERO DEI CONFRONTI



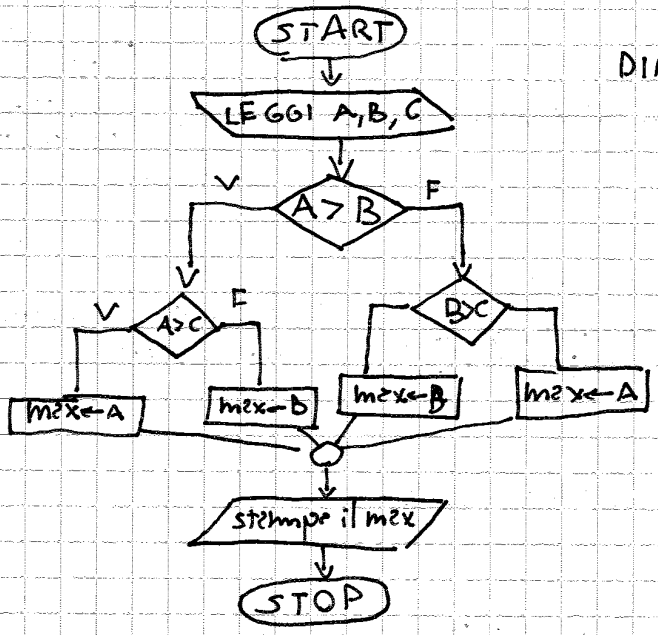


DIAGRAMMA DI FLUSSO

COSTRUZIONE DI DIAGRAMMI DI FLUSSO

Però le regole ci permettono di legare i diversi blocchi sono troppo flessibili → voglio che abbiano delle regole strutturali.

I DDF (dieg. di fluss.) LIBERI sono poco utili per programmare

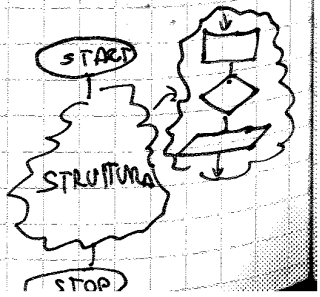
DDF STRUTTURATI → vincolati, che seguono regole ben precise → traduzioni immediate.

DDF STRUTTURATI

Nn si ragiona + in blocchi ma in STRUTTURE := particolari combinazioni di blocchi := COMPOSIZIONE DI BLOCCHI ELEMENTARI (□ ▱ ◇)

Def è un insieme predifinito di STRUTTURE ELEMENTARI:

- Un solo blocco start
- Un blocco stop
- SEQUENZA DI BLOCCHI (di azione e/o di INPUT/OUTPUT)
- if-then-(else) • while-do • ...



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 9 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 11 di 100

TIPI DI PROBLEMI

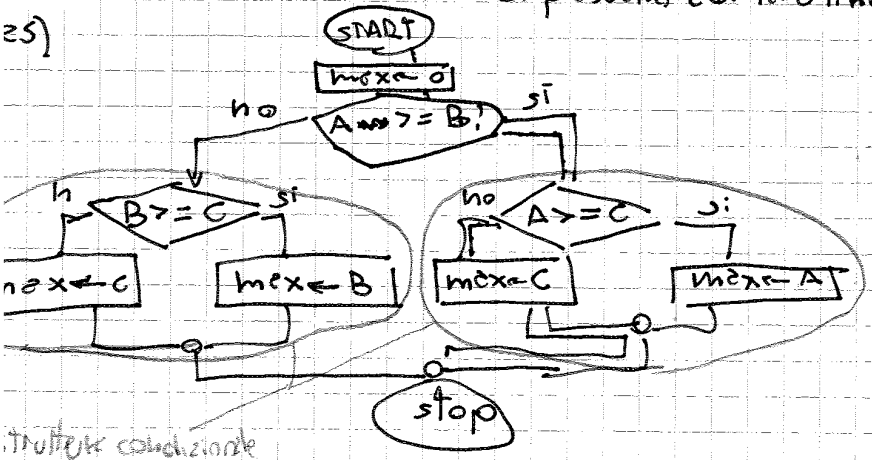
- PROBLEMI SEQUENZIALI
- PROBLEMI CONDIZIONALI
- PROBLEMI ITERATIVI

:= sequenza di operazioni

ROBLEMI SEQUENZIALI → sequenza con un solo flusso di esecuzione, senza alternative
(es. ricetta di cucina)

ROBLEMI CONDIZIONALI → c'è almeno una decisione, una operazione con più alternative
si possono contare il numero dei flussi

es)

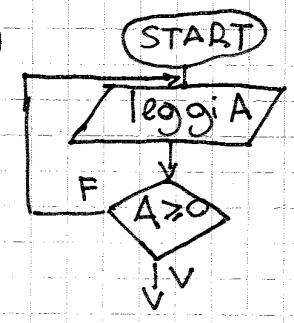


istruzioni condizionate

esemp. A?2 B?-3 c'è lo
⇒ ho 4 flussi e uno solo vale per questi A, B, C

ROBLEMI ITERATIVI → problemi dove non si possono contare le differenti vie. Contengono almeno una ITERAZIONE := ripetizione di una operazione per un certo numero di volte

es)



finché metto valori negativi il programma ricomincia da capo ⇒ non posso sapere quanti flussi ci sono poiché dipende dal numero dei dati che gli metterò dentro.

1) Calcola la somma S dei primi N numeri interi, dove N è letto da tastiera

2) Soluzione informale: $S = \sum_{i=1}^{N} i$

3) Soluzione informale 2:

a) $S = 1+2+3+4+5+6+7+\ldots+N$ → in informatica "..." non esiste ⇒ PROBLEMA ITERATIVO

b) x gauss $\sum_{i=1}^{N} = \frac{N(N+1)}{2}$ non consideriamo di non conoscerla

c) SOLUZIONE ITERATIVA →

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 11 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 13 di 100

DALLA SOLUZIONE AL PROGRAMMA

Si trascrive la soluzione formale in un linguaggio comprensibile al pc senza troppe difficoltà

LIVELLI AD ALTO LIVELLO:=
Ci sono diversi livelli di astrazione con complessità equivalente alla complessità dei singoli blocchi. (es. alto livello: C, C++, Basic, Pascal, Fortran, Java etc...)

ASSEMBLER:= sono linguaggi che utilizzano micro-operazioni che dipendono dalla macchina.

es. ALTO LIVELLO) ...
if (x>3) then x=x+1;
...

es. ASSEMBLER) ... LOAD A; MEM[1000] ADD A, 10 TEST A, 0 ... specifico per una singola architettura (microprocessore)

ELEMENTI DEL LINGUAGGIO

- Key word:= parole riservate → non può essere utilizzata per un altro scopo. Costituiscono i "mattoni" del linguaggio
- Dati:= visto DAL PC → insieme di bit memorizzato
  visto DA NOI → quantità associata ad un certo livello. È individuato da: - Un nome (identificato
  → una interpretazione →
  → come vengono interpretati i bit. (il tipo)
  - modalità di accesso
  → posso o non posso modificare i miei dati

Ogni dato viene memorizzato e la posizione dove è memorizzata è detto INDIRIZZO

- ISTRUZIONI: ci sono 3 tipi di istruzioni: - PSEUDO-ISTRUZIONI (non eseguibili)
  - ISTR. ELEMENTARI: operazioni
  - ISTR. CONTROLLO FLUSSO:

Manca un pezzo

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 13 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 15 di 100

77°00' − 5°24' + 2°26' = 74°02'

R_V = 78° [illegible] V_e = 9 Kms $S = 9 + 9,9 = 18,9$ km

$t = \frac{18,9}{9} = 2,1^h = 2^h06'$

Arrivo [illegible] = 8:00 − 2h06' = 5:54

$S_{PRIS} = 6,3$

$Rlb_I = 332° + 5,24' + 2°36' = 334°12'$ → 334

$Rlb_{II} = 73° - 5,24' + 2°36' = 70°12'$ → 70

$S_{PRIS} = 6$ km $S_1 = 2,5 \cdot 0,92 = 2,3$ km $V_{OR} = \frac{2,3}{0,77} = 3$ Kms

$V_e = 12,6$ Kms $S = 20$ km $t = \frac{20}{12,6} = 1,59 = 1^h\frac{35}{60}$ → 2h [illegible] = 11:06

R_V = 131°00' − 2°36' = 128°24' [illegible]

[illegible]

Rlb_I = 255 + 13' Rlb_{II} 228

↓ +2°26' +1°18'

258°44' 231°49'

↓ ↓

259° 232°

R_V = 329° S_2 = 4 km V_p = 18

$S_{PRIS} = 0,92 \cdot 18 = 7,6$ $S_2 = 0,2$ $V_{OR} = \frac{0,2}{0,92} = 1,9$ Kms

S = 18,9 R_V = 33°00' − 2°26' = 30°34'

↑

R_V' 37°34'

V_e = 16,8 Kms $t_{PRIS} = \frac{18,9}{16,8} = 1,13 \rightarrow 1^h08'$

S = 10,7 R_V = 17°00' − 2°26' = 14°34' + 6°42' = 21°22'

V_e = 16,5 Kms $t = \frac{10,7}{16,5} = 0,65 \rightarrow 39' \rightarrow 10:12$

S_{PRIS} = 7,9 S = 25,1 $t = \frac{25,1}{18} = 1,39 \rightarrow 1^h23' \rightarrow 9:58$

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 15 di 100

10. Una nave a vela che naviga con mura a dritta, deve dare precedenza ad una nave con mura a sinistra?
- [x] No;
- [ ] Si;
- [ ] Si, se a vele quadre.

11. A che cosa è dovuta la pressione atmosferica?
- [ ] Allo rotazione terrestre;
- [x] Al peso dell'aria che sovrasta un determinato punto;
- [ ] Attrazione del solo e della luna.

12. Quale, delle sotto elencate navi, non è considerata in navigazione secondo il Regolamento per evitare gli abbordi in mare:
- [ ] Nave ferma e senza abbrivio;
- [x] Nave incagliata;
- [ ] Nave in retromarcia.

13. Che cos'è un bollettino meteo?
- [ ] Una comunicazione che descrive il tempo in atto su certe zone;
- [ ] Una comunicazione che annuncia l'arrivo di burrasca:
- [x] Una comunicazione periodica che fornisce le previsioni del tempo in zona.

14. Che cos'è la latitudine?
- [x] È l'arco di meridiano compreso tra il punto ed il polo;
- [x] È l'arco di meridiano compreso tra il punto e l'equatore;
- [ ] È l'arco di parallelo compreso tra il punto ed il meridiano fondamentale.

15. La tabella delle deviazione della bussola, è un documento ufficiale?
- [x] Si;
- [ ] No;
- [ ] Si, per le navi di lunghezza superiore a 50 m.

16. Che cosa è l'angolo di prora o Prora?
- [ ] È l'angola compreso tra la direzione del parallelo e la prora;
- [x] È l'angolo compreso tra il meridiano ed il piano trasversale della nave;
- [x] È l'angolo compreso tra la direzione del nord e la direzione della chiglia.

17. Come si riconosce la stella polare?
- [x] Prolungando la congiungente delle stelle poppiere del carro dell'orsa maggiore del lato apposta all'inclinazione del timone e riportando 5 volte la distanza tra queste due stelle;
- [ ] Prolungando la congiungente delle stelle poppiere del carro dell'orsa maggiore del lato apposta all'inclinazione del timone e riportando 4 volte la distanza tra queste due stelle;
- [x] Prolungando la congiungente delle stelle poppiere del carro dell'orsa maggiore del lato apposta all'inclinazione del timone e riportando 3 volte la distanza tra queste due stelle.

18. Che cosa è un luogo di posizione?
- [ ] È un sistema per ottenere il punto nave;
- [ ] È una linea lungo la quale deve muoversi la nave;
- [x] È una linea lungo la quale si trovano tutti gli osservatori che effettuano la stessa misura.

19. Quale, dei seguenti, non è un componente dell'ecoscandaglio
- [ ] Trasmettitore;
- [ ] Ricevitore/Amplificatore;
- [x] Ripetitore.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 19 di 100

es) operazioni tra reali $h \cong 10 \rightarrow$ 10 cicli
operazioni tra interi $h \cong 1 \rightarrow$ 1 ciclo
confronto $h \cong 1 \rightarrow$ 1 ciclo
input-output $h \cong O(1000) \rightarrow$ 1000 cicli
memorie ?

MIPS (Million Instruction per clock)

- f = frequenza del clock [Hz = cicli/s]
- T = periodo per clock = 1/f [s]
- C = cicli macchina / istruzioni considerando una media pesata.
- IPS = f/C = 1/(T·C) [istruzioni al secondo → istr./s]
- MIPS = IPS/10^6 [istr./s] es) Umano: < 10^6 MIPS (1 oper. al secondo)

- C = 2 cicli per istruzione (in media)

se ho un pc che va a 1 GHz

$\rightarrow$ IPS = $\frac{1\,GHz}{2\,cicli/istr.} = \frac{10^9}{2} = 5 \cdot 10^8$ IPS $= 5 \cdot 10^2$ MIPS = 500 MIPS

$\Rightarrow$ un calcolatore è in media un milione di volte + veloce di noi

Riassumendo

sistema di elaborazione: Units di I/O + memoria + processore

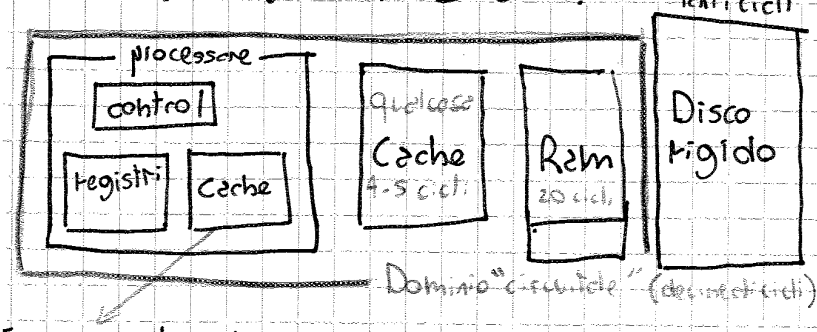
schema generale: il processore preleva istr. dalla memoria e le esegue

(micro) processore:
- Unità operativa: - ALU
  - FPU
  - Registri
  - Flag (registro che contiene informazioni di stato)
- Units di contro: - Muove il ciclo base
  - Usa PC e IR per mantenere lo stato

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 19 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 21 di 100

⇒ memoria: 1 + € cicli.



è un'appendice della cache → 1 ciclo

Memorie RAM

2 tipi: • RAM STATICHE (SRAM): - veloci
- piccolo impaccamento
- elevato costo per bit

È integrata nel processore (cache)

• RAM DINAMICHE (DRAM): - almeno 1 ordine di grandezza + lenta delle SRAM
- impaccamento + grande
- costo per bit abbastanza + piccolo

Ora si utilizzano di + le SDRAM e le DDR2-DDR3

memoria Flash

È la memoria di tutte le schede di memoria (SD, chiavette USB). Sono memorie che permettono di avere una via di mezzo tra il disco rigido e le cache.

Sono memorie non volatili con velocità simili a quelle delle RAM, ma costo ancora un po' (8GB → 20 €)

Sono le evoluzioni delle memorie ROM (Read only memory) che si potevano solo leggere senza modificarle.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 21 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 23 di 100

DALLA SOLUZIONE AL PROGRAMMA

Bisogna tradurre : diagramma di flusso nel linguaggio C

LINGUAGGIO C

È un linguaggio:
- Imperativo ed alto livello (non ci sono le micro operazioni)
- è strutturato, mecanismo delle eccezioni
- tipizzato: ogni oggetto deve avere un tipo
- elementare: ha poche Key word
- case sensitive: c'è differenza tra maiuscolo e minuscolo
- portabile
- standard ANSI: è comune per tutti → segue standard internazionali

Esempio)
```
# include <stdio.h>          → programma minimo
int main (void)
{
    print (" hello, world\n");
    return 0;
}
```

Applicazione "console"

SCRITTURA DEL PROGRAMMA → FILE SORGENTE → COMPILATORE → PROGRAMMA ESEGUIBILE

↓

Blocco note

scrivo il programma → prog.C → compilatore → prog.exe ⇒ separo le 2 fasi

Ambienti integrati

IDE (Integrated Development Environment) → noi usiamo Codeblock (blocco note)

⇒ le 2 fasi non sono completamente separate.

Tasti principali:
- settare debug
- build and run → compile e lo lancia

N.B
ALT + 123 → {
ALT + 125 → }

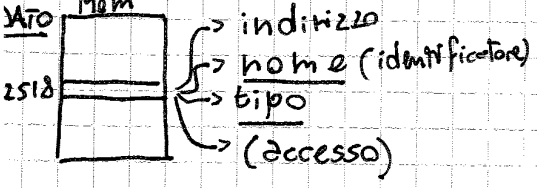
© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 23 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 25 di 100

② DICHIARAZIONE DI DATI.

Si fa all'inizio, dopo le graffe e vale fino alle prossime graffe

DATO



TIPI: sono limitati. 4 tipi fondamentali:

- char caratteri ASCII (tasti della tastiera) — blu → parole chiave
- int interi (complemento a 2)
- float reali (floating point singola precisione)
- double reali (" " doppie ") — utilizza + numeri → + spazio

l'int, il float e il double hanno dimensione variabile e secondo del processore

|char| = 8 bit = 1 byte

es) unsigned short int → → intero corto senza segno

sono previsti dei modificatori per i tipi

modificatori di segno:

- signed/unsigned → su char e int
  - signed ↓ valore con segno
  - unsigned ↓ valore senza segno → + spazio

modificatori di dimensione:

- short/long → su int
  - short ↙ occupa poco spazio → intero piccolo
  - long ↘ occupa tanto spazio → intero grande

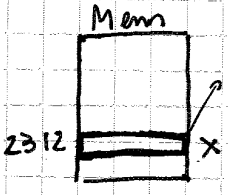
Solitamente signed è sottointeso → int e char di base hanno segno

La sintassi per definire una VARIABILE (= un dato) è:

<tipo> <identificatore> ; → l'istr. è finita

Oppure: <tipo> <lista di identificatori> ;

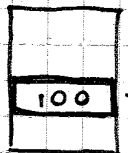
es) int x; char; long int x1, x2, x3; double pin; short (int) stipendio
long y, z; int valore;
↓ è un nome che do ~~modifico un valore~~ x indicare un valore



è uno spazio a me riservato dove metto il nome x. È uno spazio libero nella memoria

ACCESSO Si può BLINDARE una VARIABILE: nn posso + modificare quella variabile

→ const int X = 100; X è = 100 e nn può + essere modificato



Per convenzione le variabili costanti si scrivono in maiuscolo in modo da distinguerle con le variabili (es) const int float ALIQUOTA = 0,21)

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 25 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 27 di 100

⇒ printf ("....", ...)

ci sono 2 tipologie di informazioni: a) testo libero → appare come lo scrivo

b) direttive di formato → sono fatte da: %"lettera"

⇒ -%d : int → stampa un intero

-%c : char → " " carattere

-%f : float → " " reale

es) int x=2;
float z=0,5;
char c='a';
printf("%d %f %c/n", x, z, c); ⇒ 2 0,5 a

N.B
/n → ritorno a capo
:= backslash n

è ordinata secondo i tipi di caratteri che deve stampare. se l'ordine è sbagliato mi dà errore

printf("%f***%c***%d/n", z, c, x) ⇒ 0,5***a***2 (solo x per scelta)

SCANF (scanf)

scanf (<formato>, <arg1>, ..., <argn>) scansione formattata

scanf ("...", ...)
STRINGA DI FORMATO — LISTA DI VARIABILI*

le scanf è una ASSEGNAZIONE DINAMICA := il cui valore non è scritto nel programma (statico) ma è scritto dall'operatore (dinamica)

STRINGA DI FORMATO: ci sono solo direttive di formato. Se scrivo del testo libero nella scanf, questo testo deve essere "consumato"

LISTA DI VARIABILI: il suo indirizzo → le & (e commerciale) indica l'indirizzo del valore che metto

es) scanf("%d", &x); ⇒ prendi x da tastiera

printf("Il valore di x è: %d/n", x) ⇒ stampa: "Il valore ... x è: x

Prima di una scanf conviene mettere un printf con scritto "dammi un valore intero" in modo da non mettere valori reali o simili.

Per mettere + valori basta fare scanf("%d%d", &x, &y") e nel compilare ci sono 3 separatori standard → SEPARATORI DI INPUT:

- spazio (i) • tab (→) • invio (ritorno a capo) Se non uso questi il pc non lo capisce

se metto scanf("%d**%d", x, y) mi vengono risultati sbagliati

Per farlo funzionare scrivendo es) 5**4 ⇒ mi dà il risultato giusto

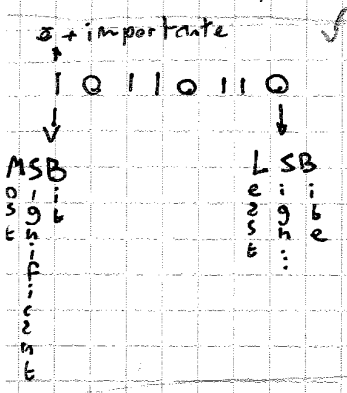
Gli ** devono essere consumati

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 27 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 29 di 100

Visto che il bit è molto piccolo si tendeva raggrupparli:

- Byte → 8 bit
- Word → n byte



→ int x = 2345724578101 7...;

se lo stampo mi modifica il numero trasformandolo in un numero + piccolo

int = 32 bit → se n = 32 che numeri ci stanno?

In base 10 con 3 cifre: max? 999 → da 0 a 10^3-1; quanti? 1000 → 10^3

In base 2 con 3 bit: max 111 → 7 da 0 a 2^3-1; quanti 8 (da 0 a 7) → 2^3

⇒ n bit → 2^n valori (0, 2^n-1) se ho 32 bit → 2^{32} valori = 4294967295 = 466 → massimo valore che raggiungo

N. B
Una parte di memoria riservata x i numeri >0 e una parte x quelli <0

long int = 64 bit → 2^{64} valori

Quindi se vado oltre i valori che riesce a memorizzare → mi dà un numero sbagliato

es) 11011_2 (= 27_{10}) n = 4 bit → 1011 → 13

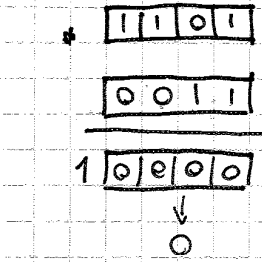
SOMMA IN BINARIO

```
 1 1 1 1 0 1 +
   0 0 1 1
-----------
 1 0 0 0 0
```

le combinazioni sono: 0+0=0; 0+1=1; 1+0=1; 1+1=10_2 → 0 (↘ 1 riporto 1)

```
  1 1 1 2
    1 3
-------
  3 0
```

N.B se sommo 2 numeri e mi viene un numero sbagliato potrebbe essere xché non c'è + spazio



n = 4 → $2^4 = 16_{10}$ valori (0 ..... 15_{10})

13_{10} + 3 = 16_{10}

:= OVERFLOW → il risultato non è rappresentabile con quel numero di bit: non c'è spazio

Non si può mai verificare quando si sommano 2 numeri con segno discorde

Avviene quando 2 numeri con segno uguale danno un risultato con segno discorde.

A volte le cifre binarie, per comodità, vengono raggruppati in gruppi di 3 o 4.

⇒ base ottale (0,1...8) o esegesimale (0...9, A...F) (es. FA05 → 16 bit)

$8 = 2^3$ raggruppo 3 bit alla volta

$16 = 2^4$ B=16 16 cifre (0...9, A...F)

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 29 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 31 di 100

32-bit → $\sim \pm 9\,999\,999 \cdot 10^{\pm 38}$ 64 bit → $\sim 9\,999\,999\,999\,999\,999 \cdot 10^{\pm 308}$ ⇒ floating point (precisione variabile)

Elaborazione dell'informatica non numerica

se ci sono K oggetti (es. caratteri alfanumerici) ho bisogno di n (2^n) bit. $n = \lceil \log_2 K \text{ bit} \rceil$ (arrotondato all'intero superiore)

gli alfanumerici sono circa 100 → K=100 n = 7 bit Ad ogni carattere binario associato un numero. ⇒ TABELLA CODICE ASCII ⇒ char es) char c = a; → 8 bit (ho 2^8 bit)

- 0 ... 31 → controllo
- 32 ... 47 → interpunzione ⇒ c'è un certo ordine
- 48 ... 57 → cifre.
- etc...

↓

156 caratteri (gli ultimi 128 caratteri codificano alfabeti "regionali" (non standard)

Quindi per es) "Ciao" ho la C come il simbolo Ascii associato alla maiuscola (67); stessa cosa per gli altri caratteri

es) char c_1 $c_1 = 'a'$; → 97$_{10}$ → ( )$_2$ binario per 8 bit → 01100110 → Ottetto c_1

printf ("%c %h\n", c_1) ⇒ a (97) → il n° del carattere Ascii

Se pongo un'altro $c_2 = '+'$ (43 Ascii)

$c_1 + c_2$ ⇒ i (CA2 → d. 1) (-116 numerico)

se gli dico printf ("%c" c_1+1) ⇒ b

CODICE UNICOD → esprime tutti i caratteri di tutte le lingue del mondo (+di un milione)

UTF-8 è la codifica unicode + usata.

---

b Espressione

<variabile> = <espressione> → operandi e operatori : - operatori aritmetici

tipo ⟷ tipo

- operatori relazionali (di confronto)
- " logici
- " di modifica del tipo (:= cast)
- " di calcolo delle dimensione di un tipo: sizeof()

Operatori aritmetici

Sono 4: +, -, *, /

N.B 2/3 2.0/3.0 (circuiti)

Se metto 2.0/3.0 costringo il calcolatore a fare un operazione con i reali ← stesso simbolo per operazioni diversi → se metto 2/3 il calcolatore fa un operazione con i numeri interi

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 31 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 33 di 100

`+=` → `x+=3;` ⇒ `x=x+3` `x+=y;` ⇒ `x=x+y`

`-=` → `x-=3;` ⇒ `x=x-3`

OPERATORI LOGICI

Combinano delle operazioni fornite espr. booleane e forniscono un risultato "booleano"

`!` `&&` `||`
`not()` `()and()` `()or()`

es) AND: `(x>=a) && (x<=b)` ⇒ $a \leq x \leq b$ se $a<b$
$x \leq b$ $x \geq a$ se $b<a$

OR: `(v1>=18) || (v2>=18)` ⇒ • o v1 o v2 deve essere ≥ 18

NOT: `!(a>b)` ⇒ non $a>b$

es) `int a=5 b=3 c= a>b` se $a<b$

`printf("%d",c)` ⇒ $1 \neq 0$ → Vero ⇒ 0 → Falso

... `c != (a != b)`

... ⇒ 1 → Vero

LOGICA DEGLI OPERATORI ELETTRONICI

Logica di Boole: G.B. introdusse una logica basata su enunciati che potevano essere veri o falsi.

Le variabili possono assumere solo 2 valori: - vero
- falso

Bisogna distinguere le variabili dipendenti e indipendenti

$x \in \mathbb{B} = \{V, F\}$ $A \in \mathbb{B} = \{V, F\}$ $B \in \mathbb{B} = \{V, F\}$

A	B	A operazione B
F	F	F
F	V	F
V	F	F
V	V	V

:= TAVOLA DELLA VERITÀ

Operatore not !() (negazione di A)

A	$\bar{A}$ ($\neg A$)
F	V
V	F

Operatore OR ()||()

A	B	$A \overset{or}{\vee} B$ ($A +^{+} B$)
F	F	F
V	F	V
F	V	V
V	V	V

Operatore AND ()&&()

A	B	$A \overset{and}{\wedge} B$ ($A \times B$)
F	F	F
F	V	F
V	F	F
V	V	V

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 33 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 35 di 100

OPERATORE SIZEOF

Ci dice le dimensioni di un <tipo> in bit

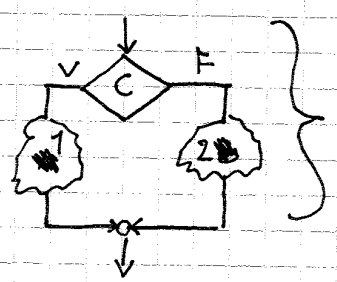
sizeof (<tipo> oppure <espressione>) ⇒ /* size=4 /*

N.B
Quando copio e incollo da word devo stare attento alle virgolette (" nn funziona; deve essere ")

ISTRUZIONE IF

ci sono 2 condizionali: • if ... else
• switch ((scelta)

STRUTTURA CONDIZIONALE



IF ... ELSE

```
if (<condizione>)
{ <blocco1>          --> V
[{}else
{ <blocco2>}]        --> F
}
```

può non ci essere l'else

1 e 2 sono delle strutture + o- complesse. Se 1 o 2 consistono di una sola istruzione ⇒ questa sintassi è sufficiente. Se 1 o 2 consistono di + istruzioni ⇒ si usano delle graffe {1} {2} (cmq. sarebbe meglio usarle sempre)

N.B. Noi possiamo eseguire o il blocco 1 o il blocco 2, c'è un salto di quale riga

• es) calcolo del modulo di un numero $|x| = \begin{cases} x & \text{se } x \geq 0 \\ -x & \text{se } x < 0 \end{cases}$

N.B
/* ... */ := COMMENTI
→ il calcolatore non lo conta, serve solo a chi legge il file nel compilatore

```
/* programma che calcola il valore assoluto di un numero intero
   letto da tastiera */
int N, va;
scanf ("%d", &N)
if (N>=0)
{
   va=h;
}
else
{
   va=-N;
}
printf("valore assoluto: %d\n", va);
...
```

Si può anche scrivere così

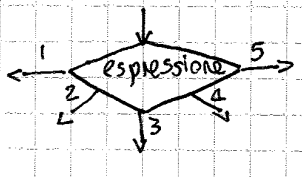
```
int N, va;
scanf("%d", &N)
va=N
if(N<0)
{ va=-N
}
printf ("valore assoluto: %N", N)
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 35 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 37 di 100

ISTRUZIONE SWITCH

Ci permette di discriminare tra varie possibilità




```
switch (<espressione>)
{
   case <costante1>:        1° caso:
        <blocco1>
        break;
   case <cost.2>:           2° caso:
        <blocco2>
        break;
   ...
   default:
        <blocco default>
}
```

serve quando per esempio c'è un errore, quando non si verifica nessuna condizione

es)
```
int a;                  printf("premi 0 per la 1° opzione")
scanf("%d", &a);                  1 " " " "
                                  2 " " " "
switch (a)
{ case'0': printf("A=0\n"); break;
  case'1': printf("A=1\n"); break;
  case'2': "    ("A=2\n"); break;
  default: "    ("Il valore non e' ne' 0, ne' 1, ne' 2");
}
```

Se metto o tolgo scanf

senza break esegue poi il case 0 (se si verifica) e poi gli altri

qui possono non mettere il break xché tanto è l'ultimo e sotto non c'è niente

Se vuoi fare in modo che si esegua la stessa operazione per due caratteri (es. a e A) inseriti con la scanf devi fare

```
...
case'a':
case'A':
   ...
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 37 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 39 di 100

ISTRUZIONE FOR (vedi anche slide)

For (<inizializzazione>; <condizione>; <incremento>) Nn c'è ;
{ <blocco> } I campi possono essere vuoti

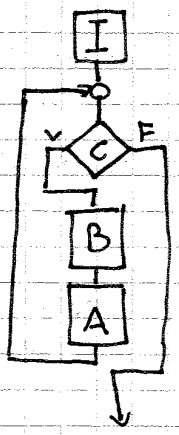
INIZIALIZZAZIONE: condizione iniziale prima del ciclo

CONDIZIONE: una condizione booleana

INCREMENTO: incremento della variabile di conteggio

Molto simile al while. (esempio di prima) Mentre nel while nn sappiamo quante iterazioni facciamo, nel for sappiamo esattamente quante iterazioni facciamo (es 2-3 di prima)

for (I; C; A)
{ B; }



è equivalente a: <inizializzazione>
while (<condizione>)
{ <blocco>
<incremento> }

es) leggi un carattere ch (es *) e stampalo N volte (es 3 → ***)

```
... int N;   scanf("%d", &N);
    char ch; scanf("%c", &ch);
    int i;   i=0
    while (i<(<=) N)
    { printf("%c", ch);
      i++; }
```

oppure

```
for(i=0; i<(<=)N; i++)
{ printf("%c", ch) }
```

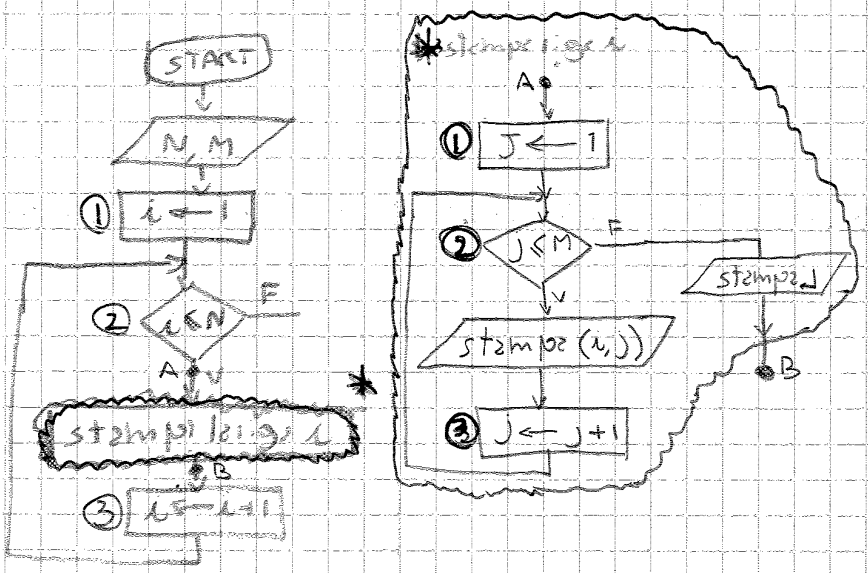
es) Di 100 num da tastiera, considerando solo quelli tra 0 e 30, determinare max e min.

```
... int num, i, max_num, min_num, num_validi;
    max_num = -1000   parto da -1000 così sono sicuro che mi darà max > -1000
    min_num = 3000    " " 3000 " " " " " " " min < 3000
    num_valido = 0    → n° dei numeri validi
    for (i=0; i<10; i++)  → i che parte da 0 e va fino a 10 incrementando di 1 ⇒ 10 volte l'operazione
    { scanf("%d", num);
        if (num >= 0 && num <= 30)  → considera solo quelli compresi tra 0 e 30
        { if (max_num = num)
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 39 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 41 di 100

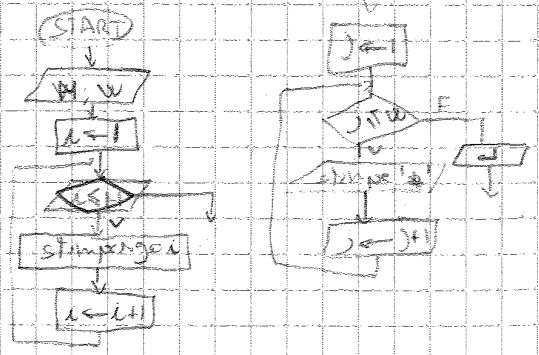
N, M
↓
stampa N righe. riga i: (i,j) j=1..M
i=1..N




```
...
int i, j, N, M;
printf("inserire n° righe e colonne");
scanf("%d%d", &N, &M);
for (i=1; i<=N; i++)
{
   /* stampa la riga i */
   for(j=1; j<=M; j++)
   {
      printf("(%2d,%2d)", i, j);
   }
   printf("\n");
}
...
```

) H, W da tastiera, stampa un rettangolo HxW di asterischi

```
...
int i, j, H, W; char ch;
scanf("%d%d", &H, &W);
scanf("\n%c", &ch);
for(i=1; i<=H; i++)
  { for(j=1; j<=W; j++)
      { printf("%c", ch); }
  printf("\n"); }
...
```



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 41 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 43 di 100

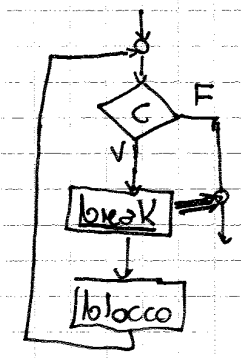
Es) leggi N controllando che il valore sia positivo. In caso contrario ripetere la lettura.

```
...
int n;
do                                  Ripeti la scansione
  {scanf("%d",&n);}
while (n<=0);                       finché n ≤ 0
...
```

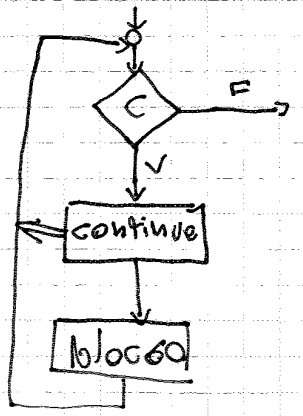
N.B

```
int n;
scanf("%d,&n)
while (n<=0)
  {scanf("%d",&n);}
...
```

Ci sono due istruzioni che possono modificare il normale svolgimento del flusso:

- break: 

ti fa uscire dell'istruzione e ti porta sull'altra parte

- continue: 

ti fa saltare un blocco e ti fa ritornare alla condizione. Si può sostituire con l'if

Sia break che continue si possono SEMPRE non usare

Noi NN LO USIAMO MAI!! (xché in C esiste ma in altri linguaggi non è detto

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 43 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 45 di 100

VETTORE := insiemi di variabili dello stesso tipo aggregate in un unico operazione

Sintassi: <tipo> <nome vettore> [<dimensione>]; (NO) int v[x];

<dimensione> deve essere costante (non può essere una variabile come x o y)

Accesso ad un elemento: <nome vettore> [<posizione>]

es) int v[4] → 4 variabili intere ⇒ v[0], v[1], v[2], v[3]

int dato [3]

tipo degli elementi — nome del vettore — dimensione

N.B [<dimensione>] deve essere un numero intero e positivo. Se do seguo questa regola v[ ] → qui posso mettere ogni tipo di espressione

es) double a[100];
double x;
int i, j, K;
... ...
a[3+i+j+K]

I cicli sono utili per scandire un vettore

es) int v[10000];
int i;
/* riempio il vettore (tastiera) */
for (i=0; i<10; i++)
{
v[i] = v[i]/2;
}

al posto di

v[0] ← v[0]/2
v[1] ← v[1]/2
...
v[i]

Inizializzazione di un vettore

con int posso fare int x = 2; → questo è uno scalare

con i vettori → • int [4] = {2, -1, 5, 10} → insieme di valori

2	-1	5	10
v[0]	v[1]	v[2]	v[3]

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 45 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 47 di 100

Copia di un vettore:

```
for (i=0; i<N; i+1)
{ w[i] = v[i]
}
```

es)
```
... → #define N 5
int v[N] = {-2,4,1};
int i;
for (i=0; i<N; i++)
{
  printf("v[%d] = %2d\n", i, v[i]);
}
```

$\Rightarrow$ v[0] = -2, v[1] = 4, v[2] = 1, v[3] = 0, v[4] = 0

int v[N] = ", w[N]

* se metto w = v → errore. ↓ Se stampo w mi dà dove w[0] è stato memorizzato e subito dopo ci sarà w[1] etc...

```
#include N 5
...
int v[N] = {-2,4,1}, w[N];
int i
for(i=0; i<10000; i++)
{
  v[i] = -1;
}
```

ERRORE

io ho messo v[5] che occupa 5 spazi in memoria se eseguo invado la memoria dove non dovrei andare e quindi mi dà errore.

) int v[5]

v[5]=0 ERRORE xché io ho solo dà v[0] à v[4] e quindi in vado uno spazio non mio. Può accadere che lo spazio dopo v[4] (il v[5] che cerco di usare) è occupato da un'altra variabile e quindi assegno un valore alla mia variabile senza volerlo

V.B V = 0 UN VETTORE NN STA MAI A SX DELL'UGUALE

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 47 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 49 di 100

Ricerca di un elemento in un vettore

Dato un valore numerico, verificare: - se almeno uno degli elementi del vettore è uguale al valore numerico
- in caso affermativo, dire dove si trova o dire che non esiste

→ RICERCA DI ESISTENZA

$x \in v[]$? per ogni $v[i]$ $x = v[i]$? Si o no.

```
#define N 10...
... int dato;   /* dato da ricercare */
    int trovato; /* flag per la ricerca */
    int v[N] = {2,3,5,-1,-2,0,4,7,28};
    int i;      /* posizione elemento */
    printf("Elemento da ricercare? ");
    scanf("%d", &dato);
    trovato = 0;
                          così quando lo trovo esco dal ciclo.
    for (i=0; i<N && !trovato; i++)
    {
        if (v[i] == dato)
        {
            printf("valore trovato...");
            printf("in posizione %d\n", i);
            trovato = 1;
        }
    }
    /* del ciclo sono uscito se:
       a) i >= N   b) trovato = vero
    */
    if (!trovato)
    { printf("Elemento non trovato"); }
```

N.B

A && B → vero → resta nel ciclo

!(A && B) → esco dal ciclo quando → !A || !B

(De Morgan)

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 49 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 51 di 100

Ricerca di esistenza o universalità

È vero che (nessuno) degli elementi verifica la proprietà? $\forall x : (!)\, P(x)$

È vero che almeno uno (non) verifica la proprietà? $\exists x : (!)\, P(x)$

Esistenza: $\exists x : !P(x)$

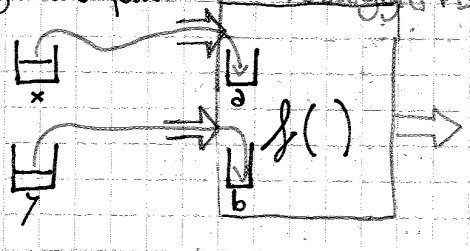
- Inizializzo flag F=0
- Ciclo su tutte le x:
  se P è vera (False): F=1
- Se F=1 l'esistenza è verificata
- Se F=0 " non " "

Universalità: $\forall x : !P(x)$

- Inizializzo il flag F=1
- Ciclo su tutte le x:
  se P(x) è falsa (vera): F=0
- Se F=1 l'universalità è dimostrata
- Se F=0 " " non " "

Manca un pezzo

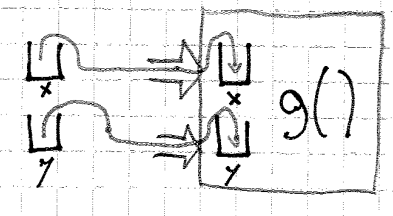
Passaggio dei parametri: Passaggio PER VALORE



r = f(x,y) INVOCAZIONE

```
int f(int a; int b);
```

Passaggio per INDIRIZZO

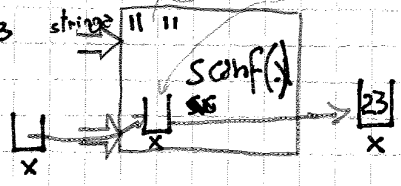



```
main()
{? x, y;
   g(x,y)   // x e y hanno un valore
            // x e y sono cambiate
}
```

Voglio che nn definisca x e y. g() conosce e può usare x e y. Voglio che x e y possano variare.

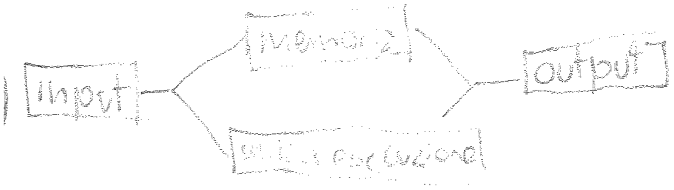
Nella `scanf("%d", &x);` ci permette di cambiare la variabile x. È una funzione

es. x=23



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 51 di 100

$(3byte + 10 \cdot 4 = 67byte) \cdot 10 = 670byte$



#include

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 55 di 100

I/O

- printf("<formato>", <arg.1>, <arg.2>...) es printf("ciao %d", n)
- scanf("<formato>", &<arg1>...) es scanf("%d", &n)

- putchar(<var. (tipo char)>) es putchar(c)
- c=getchar() es while (c != EOF) c= getchar()

Leggono o stampano un carattere per volta.

- puts(<stringa>) es char s2[100] puts(s2)
- gets(<stringa>) es char s1[100] come ti chiami? gets(s2)

leggono e stampano intere stringhe (compresi gli spazi)

- c=fgetc(<file>) es c= fgetc(fp)
- fputc(<arg(tipo char)>, <file>) es fputc(c, fp)

leggono caratteri da file e stampano su file esattamente come facevano getc e putc

---

- fgets(<arg(tipo stringa)>, <n° caratteri da leggere +1>, <file>)
  es fgets(c, 50, fp) → legge 49 caratteri da fp e li mette nella stringa c
- fputs(<arg(tipo stringa)>, <file>)

come fgets e puts

- fscanf(<file>, <formato>)
- fprintf(<file>, <formato>)

come scanf e print con il file davanti.

NULL → \0 la fine della stringa. se la stringa è vuota ovvero c'è (errore) → NULL
EOF → End Of File quando finisce il file compare questo
%d → int %f → float %s → stringa %c carattere %g → double
stdin → input da tastiera stdout e stderr → output video

% [Flag] [dim. min] [.precisione] [dimensione] <carattere>
- → stampa a sx (min. dimensione stringa) (n° cifre) h → short

\n → a capo (↵)

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 55 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 57 di 100

FILES

Al posto di prendere e stampare dati da tastiera e su schermo, si può anche lavorare con i file

I file devono essere prima aperti e poi chiusi

FILE * <arg>; arg tipo file

<nome file> = fopen ("<nome file>", "<mod. apertura>")

modalità di apertura:

- "r" → sola lettura (dall'inizio fino in fondo)
- "w" → scrivi sul file cancellando ciò che c'era dentro
- "a" → append. Aggiungi in coda

chiusura file → fclose (<arg>)

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 57 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 59 di 100

FUNZIONI

Sono dei sotto-programmi

funzioni: <tipo> <nome funzione> (<argomenti>) ← lista delle variabili che raggruppano l'interfaccia della f
{ <blocco> }

es) f che somma 2 interi -> int somma (int x, int y)
{ int s;
s = x + y
return s;
}

Prototipi: le funzioni vanno dichiarate prima di essere usate
<tipo> <nome funzione> (<tipo arg.1> <tipo arg.2> ...)

es) int somma (int, int)

Procedure: quando la f restituisce risultati che devono essere ignorati anche se restituisce
Visto che il risultato è da ignorare o non esiste si usa il tipo = void (non tipo)

es) la f che stampa un vettore

```
void stampaVettore(int v[], int);
int main()
...
void stampaVettore (int a[], int n)
{ int i;
  for (i=0; i<n; i++)
  { printf ("v[%d] = %d", i, a[i]); }
printf ("\n")
return 0;
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 59 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 61 di 100

ISTRUZIONI

- if (<condizione>)
  { blocco 1 } ⇒ Vero
  else
  { blocco 2 } ⇒ Falso

- switch (<espressione>)
  { case '<caso1>': <blocco1> ; break; → caso1 come risultato dell'espressione
    case '<caso2>': <blocco2> , break; → caso2 " " " "
    ...
    default: <blocco default> → caso in cui non si verifica nessuna condizione
  }

  es) int d;
  switch (d)
  { case '0': printf("d=0\n"); break;
    case '1': " ("d=1\n"); break
    default: printf("il valore non è né 0 né 1");
  }

- while (<condizione>)
  {<blocco>}

Continua a eseguire <blocco> finché <condizione>

for (<inizializzazione>;condizione>; <incremento>)
{<blocco>}

- do
  {<blocco>}
  while (<condizione>);

ripeti <blocco> finché <condizione> è vera

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 61 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 63 di 100

Void (non-tipo, senza codifica)

Il void è introdotto nel C per poter raggruppare i valori che non hanno tipo nelle funzioni, più nello specifico nelle procedure

Non ha senso dichiarare delle variabili del tipo `void x;` !

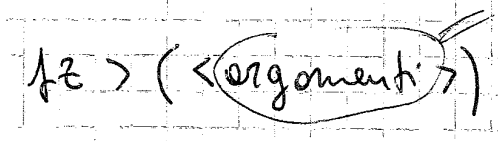
Funzioni = sequenze operative

1. **Definizione della funzione** (= cosa fa la funzione
2. **Dichiarazione della funzione** (prototipo) (= definisce l'interfaccia della funzione)
3. **Uso della funzione**

Specifica di una funzione (Definizione della fz.)

Sintassi

```
<tipo> <nome fz> ( <argomenti> )
{
   ...
}
```



Quando una fz. non ha tipo, ritorna e basta!

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 63 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 65 di 100

Funzioni e parametri

I parametri e risultato sono sempre associati ad un tipo, e vanno rispettati nell'utilizzo delle funzioni

Utilizzo della funzione

- Deve rispettare l'interfaccia della fz.
- Se la fz. non è void, la fz. va usata come generica espressione (= se prende un intero, posso associarla ad un intero, ed utilizzarla nelle espressioni) che torna un valore specificato nella sua definizione
- Utilizzata come una normale istruzione

<variabile> = <nome fz>(<{parametri}>)

- Può essere usata ovunque (può anche invocare se stessa)

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 65 di 100

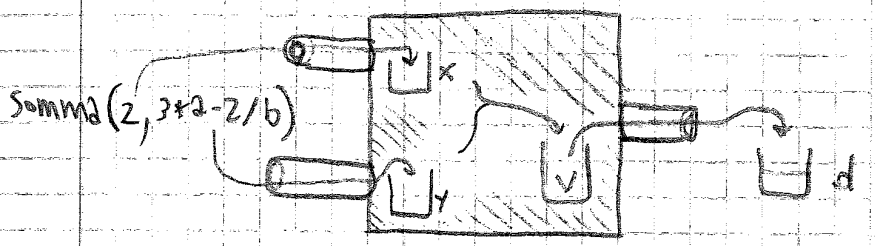
© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 67 di 100

Parametri formali e attuali

Formali: nome dei parametri in sede di dichiarazione (v1, v2)

Attuali: sono valori che forniamo fisicamente alla funzione (= generiche espressioni che hanno un certo valore).

Passaggio dei parametri (si riprenderà più avanti!)



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 67 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 69 di 100


```
return f;
}
```

coeff binomiale $\binom{x}{y} = \frac{x!}{x!(x-y)!}$

```
Copia prototipo
int main(x)
{
  int x,y  xyb;
  scanf("%od %od", &x, &y);
  xyb = fatt(x) / (fatt(y) * fatt(x-y));
}
```

int int int

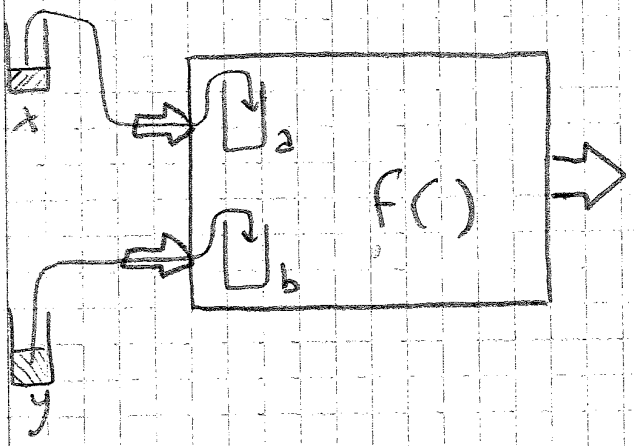
Lezione 15

Venerdì 4/05/2012

Passaggio dei parametri per valore

```
int f(int a, int b);
```

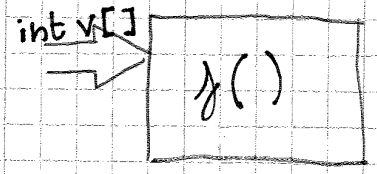
$r_{risultato} = f(x,y)$ invocazione

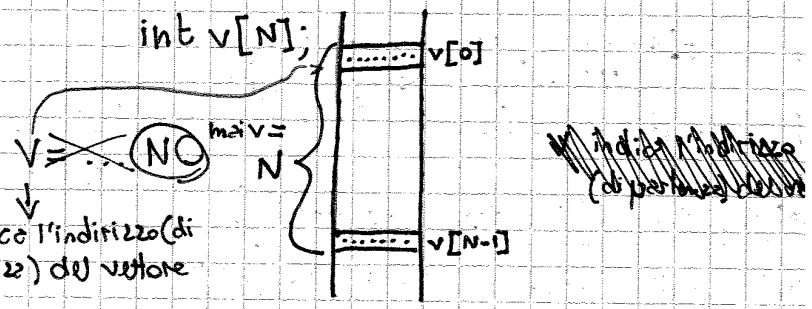


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 69 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 71 di 100

Vettori e funzioni





riempiVettore(?);

stampaVettore(?);

cercaValore(?);

vista che v è un indirizzo, io posso manipolarlo.

```
void StampaVettore(int[], int)
...
StampaVettore(v, N);
```

vettore.

→ v è di tipo int[] → così il pc capisce che è un vettore.

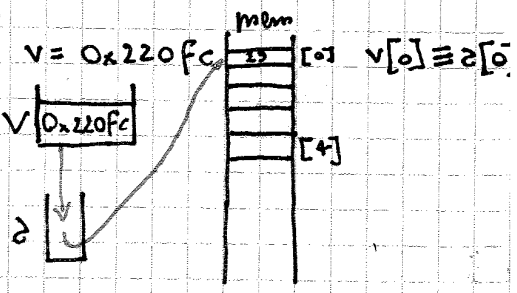
→ così capisce dove inizia il vettore (l'indir. di partenza) e dove finisce (dopo N elementi)

```
... #define N 5;
void StampaVettore(int v[], int); void RiempiVettore(int v[], int);
int main()
{
   int v[N];
   int i;
   RiempiVettore(v, N);
   StampaVettore(v, N);
   return 0
}
```

se anche scrivessi a[N] il compilatore ignorerebbe N

```
void StampaVettore(int a[], int n)
{
   int i;
   for (i=0; i<n; i++)
   {
      printf("v[%d] = %d", i, a[i]);
   }
   printf("\n");
   return 0;
}

void RiempiVettore(int a[], int n)
{
   int i;
   for (i=0; i<n; i++)
   {
      printf("v[%d] = ", i);
```



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 71 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 73 di 100

a) INIZIALIZZAZIONE

stringa

char s[10] = "abc"; → mette il NULL automaticamente quando stretto di scrivere

'a'	'b'	'c'	'\0'						

→ char t[4] = {'a', 'b', 'c', '\0'}

b) TRAMITE OPPORTUNE ISTRUZIONI di manipolazione delle stringhe:

- lettura: lettura da tastiera inserendo un'opportuna istruzione
- copia etc...

N.B: • le stringa vuota contiene sempre almeno il null.

Nelle stringhe posso anche non specificare la dim. del vettore. In questo modo viene inizializzato in base al valore assegnato. In genere è meglio mettere la dimensione

char s[] = ""; → | '\0' |

• Il carattere 'a' è rappresentato dal codice ascii ed è ≠ della stringa "a" xché "a" contiene anche il null. → "a" | 'a' | '\0' |

'a' | 'a' |

• Si possono utilizzare funzioni di lettura e scrittura di stringhe intere:

es con SCANF e PRINTF di stringhe si usa il %S)

N.B. — Per le stringhe si usa %S

char s1[10] = "prova"; ← dopo la a c'è il null

char s2[10]; scrivo 10 caratteri e poi c'è il null.

printf("cm t chiami?");

scanf("%S", s2); non si usa la & xché non è una scelta

s2[0] = tolower(s2[0]);

printf("%s", s2);

• I/O di intere righe

Legge/stampa una riga alla volta.

Le istruzioni sono: • char *gets(<stringa>): legge da tastiera e la mette nella stringa finché non preme ↵

• int puts(<stringa>): stampa una stringa

es) char s2[1000];

printf("cm t chiami?")

gets(s2);

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 73 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 75 di 100

N.B. se associo home e coghone, aghiuno con, per es., 5 spazi occupati, l'associazione hn deve superare i 5 caratteri altrimenti occupo lo spazio dove hn posso occupare.

strlen

l = strlen(str1); l ≥ 0 mi dice il n° di caratteri nulli.

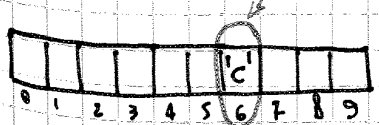
N.B Nelle funzioni con i vettori devono dargli il vettore e dove finiva (es → copiavettore(int v[], int
Nelle " " le stringhe non mi serve sapere dove farle finire xché il null le fa finire automaticamente.

strncat / strncmp / strncpy Mi dice che devo copiare, comparare o concatenare solo i primi n valori.

es)
```
#include<stdio.h>
#include<string.h>
int main()
{
  char s[80] = "stringa di prova",
       t[80], v[80];
  int i;
/* uso di strcpy */  /* equivalente di t=s */
  strcpy(t,s);        controlla che t e s sono diventati uguali → 0
  printf("%d\n", strcmp(t,s));
  strcpy(t, "Test");  ora faccio sicché t = test e v = test1
  strcpy(v, "Test1");
  printf("%d\n" strcmp(t,v)); → -1 xché t è di ordine lessicografico di un grado inferiore a s.
```

strchr

p = strchr(str, c); char *p xché strchr mi dà come risultato solamente l'indirizzo di 'c' e quindi devo per forza utilizzare un puntatore, altrimenti non riesco a sapere la posizione (es. 6)



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 75 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 77 di 100

Lezione 13

Venerdì 18-05-2012

32 bit C.2.2 → 2^{32} valori

↓

$-2^{31}, \dots, 0, \dots 2^{31}-1$

Intervallo di rappresentazione

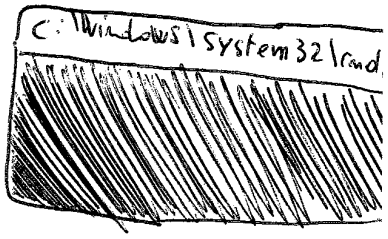
Usiamo le fz. di libreria #include <limits.h>

Slide

Argomenti sulla linea di comando

Prompt dei comandi

↘ pronto a ricevere un comando



Prima i sistemi operativi sfruttavano le linee di comando, cioè per spostarsi, ~~cercare~~ selezionare, aprire, chiudere qualcosa sul pc, ~~con lo freccia to~~ si faceva esplicitamente come siamo abituati noi: tutti gli input erano per via letteraria, cioè si dovevano digitare dei comandi (copy, open, ecc...) per fare

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 77 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 79 di 100

Esempio:

```
int main (int argc, char* argv[])
{
   int i;
   for (i=0; i<argc; i++)
   {
   puts(argv[i]); /* visualizza la stringa che si trova nella cella */
```

Usando la finestrella 'project' e selezionando "Select arguments" che funziona un po' come un watches, dove io inserirò il contenuto di stringhe, anche se contengono valori numerici.

Nel Code blocks potrò invocare il prompt di comandi, che funzionerà come in Windows.

argc e argV non sono interattivi come la scanf l'operatore immette tutti i dati e li riempie da solo

Es. C:\>prog 3 file.dat 3.223e5

Il programma la codifica scritto come una 'matrice' di caratteri.

V[0]	'p'	'r'	'o'	'g'	'r'	'\0'				
V[1]	'3'	'\0'								
V[2]	'f'	'i'	'l'	'e'	'.'	'd'	'a'	't'	'\0'	
V[3]	'3'	'.'	'2'	'2'	'3'	'9'	'5'	'\0'		

} argV

argv[1] → → '3' | '\0'

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 79 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 81 di 100

Esempio (Record dello studente)

```
char cognomi [100][80+1];
char nomi [100][80+1];
int matr[100];
float media[100];
```

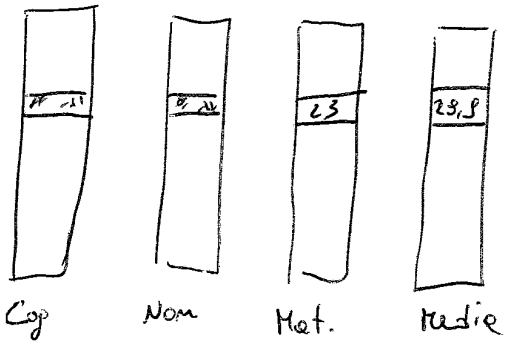
Sono 4 vettori indipendenti.

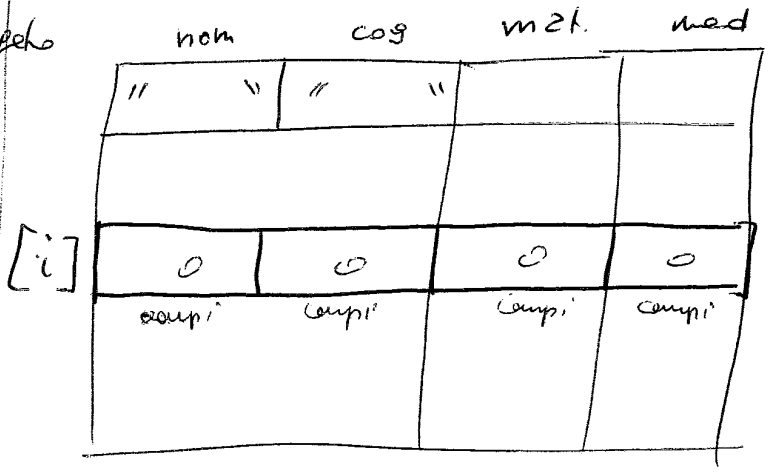
```
struct studente
{
  char cognomi 100
  char nomi 100
  int matr[100]
  float medie[100]
}
```

1 studente

```
struct studente anagrafica[100];
```

vett. raggruppato





la [i]esima riga collegata + informazioni M, C, ma, me

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 81 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 83 di 100

Accesso ad un campo della struct.

<variabile> <campo>
struct.

Si può accedere solo ad un campo per volta

strcpy(anagrafica[i].cognome, "Rossi");

Definizione di struct come tipo

Si può definire un nuovo tipo a partire da struct tramite la direttiva typedef

```
struct complex
{
 ;
}
```

def. di un tipo

prima lo si definisce, poi lo si "rinomina"

```
typedef char byte;
typedef int intero;
intero x, y;
typedef struct complex comp;
```

struct complex: vecchio nome
comp: nuovo nome

Operazioni sui struct

Di norma non si possono confrontare 2 variabili di tipo struct usando il loro nome (= vogliamo un po' il comportamento dei vettori)

[redacted]

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 85 di 100

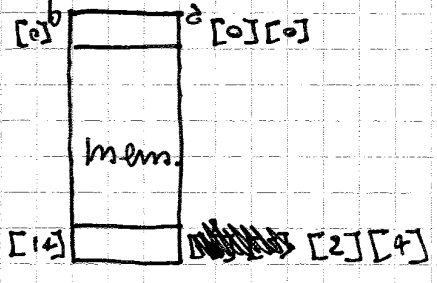
Vettori Multidimensionali (MATRICI)

Variabile indicizzata ("vettore") a più dimensioni. Si usano per creare MATRICI

int a[3][5]

matrice 3×5 = 15 interi

int b[15]



Sintassi

<tipo> <nome vettore> [<dim1>][<dim2>] ... [<dimN>];

ACCESSO AD UN ELEMENTO: <nome vettore>[<pos1>][<pos2>]...[<posN>]

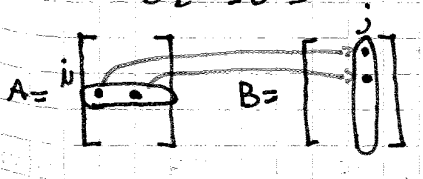
es) int v[3][2];

N.B. Per vettori a + dimensioni, la scansione va applicata a tutte le dimensioni

(vedi anche esempi su slide)

- int v[N];
  for (i=0; i<N; i++)
  { /*opera su v[i]*/ }
- int v[N][M]
  For (i=0; i<N; i++)
  {
  for (j=0; j<M; j++)
  { /*opera su v[i][j]*/ } ← opera sulla j riga
  }

es) Dato A[M][N]
B[N][P] → C = A×B = [M][P]



$$c_{i,j} = \sum_{k=1}^{N} a_{i,k} \cdot b_{k,j}$$

```
#define M 2
#define N 3
#define P 2
int main ()
{
int A[M][N], B[N][P], C[M][P];
int i, j, k;
leggi le matrici A e B
for (i=0; i<M; i++)
{
for(j=0; j<N; j++)
{ /* calcola C[i][j]*/
int s=0;
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 85 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 87 di 100

Manca un pezzo

22/05/12

FILES

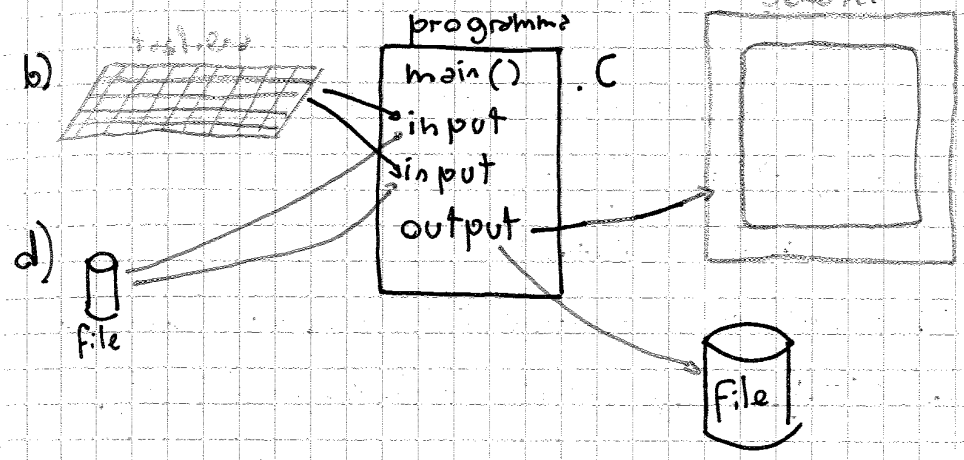
Per riempire un vettore noi possiamo:

a) inserire i valori all'inizio `int v[]={2,3...}`

b) inserire i valori da tastiera con comandi di I/O scanf, gets, getchar

c) tramite argomenti (linea di comando) argv[]

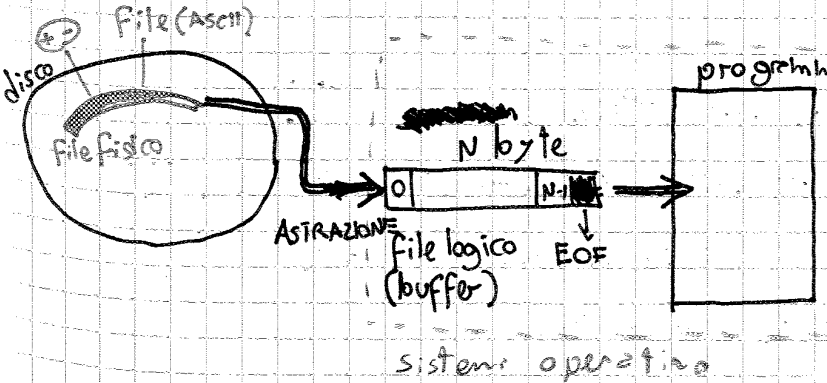
d) tramite file (b) è un caso particolare di d))



I file sono principalmente di 2 categorie: binari e Ascii (di testo)

BINARI
- FORMATO "PROPRIETARIO": non conosciamo la codifica (per es. non so a cosa servano i primi 8 numeri binari). es. .doc
- FORMATO "STANDARD": non sono file ASCII, ma conosciamo la codifica (per es. .pdf)

ASCII → sono file dove ogni 8 bit sono interpretabili come un carattere ASCII. (es. .txt, .html)



Il C vede i file come un FLUSSO (stream) SEQUENZIALE DI BYTE

I/O bufferizzato: copia di dati non presi singolarmente, ma con un insieme di dati + consistenti. Noi preleviamo dati su disco, formati da dipoli magnetici e convertiti in 1 e 0, e gli "parcheggiamo" in buffer, dove possiamo prendere i dati + facilmente.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 87 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 89 di 100

- CHIUSURA DEL FILE int fclose (FILE* <file>);

  ⟶ fclose (f);

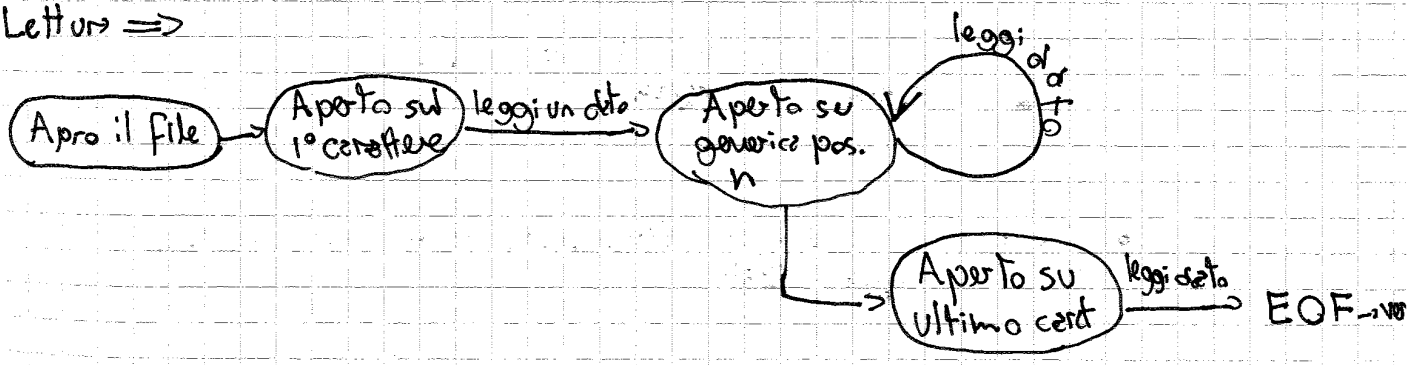
  NON LO USIAMO

es)

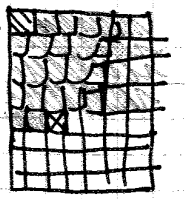
```
int n=0
FILE* fp;          va messo nella stessa cartella del file sorgente (*.c) altrimenti devo scrivere tutto il percorso
if (fp = fopen("test.dat","r");
  if (fp != NULL)
  { /* file ok, usabile */
  }
  else { printf("Errore nella lettura del file"); }
```

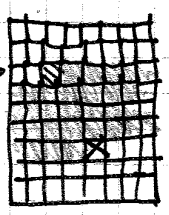
- ACCESSO AI FILE

Un file si può leggere e scrivere (in testa) o in coda). Le funzioni per leggere e scrivere sono delle varianti delle istruzioni di I/O che abbiamo già vista.



Scrittura ⟹ si divide in "w" e "a" (append)

"w" →  inizia a scrivere dall'inizio

"a" → 

I/O (standard)

a) caratteri: getchar() putchar()
b) righe: gets() puts()
c) con formato: scanf() printf()

Queste funzioni sono viste come dei file ⟹ sono dei casi particolari dell'I/O da File

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 89 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 91 di 100

C'è un file con un intero per riga e voglio memorizzarli in un vettore.

Ci sono 2 problemi: • se nn me lo dice il problema, io nn posso sapere quanto è grande il file e, nel dichiarare la lunghezza del file, rischio di occupare tanto spazio x nulla.

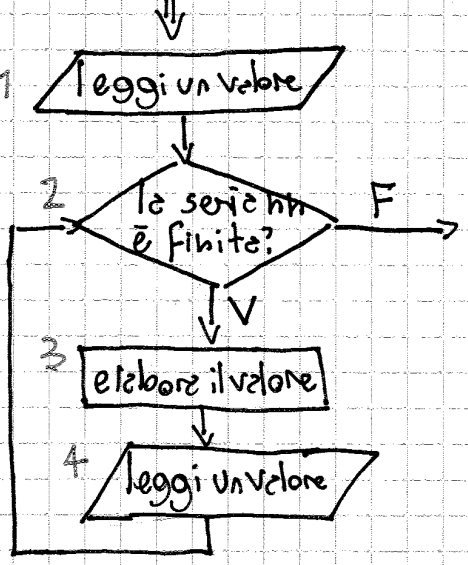
int v[N]

• supponiamo che ci possono essere al max 100 valori:

a) N=100 (ci sono 100 valori) →

→ for (i=0; i<100; i++)
{fscanf(F, "%d", &v[i]);} → faccio 100 scanf

• b) Nn so quanti valori sono → servono schemi di lettura che usano EOF, altrimenti rischio di fare 100 scanf quando ne bastavano 10 → casino



come si traduce "fineserie" nella lettura del file?

1) leggi il file
2) while (nn è finito il file) → La condizione "nn è finita il file" si può realizzare in 2 modi:
{
3) Elaboro dato
4) leggi un dato del file
}

• Usando i valori restituiti dalle funzioni di input (scanf, gets)
• Usando la funzione feof()

Valori di ritorno delle funzioni di input

• = fscanf(F, "%d%d%d", &x, &y);
↳ int / se la scanf riesce ≥0 → 3 (ci sono 3 %d e se tutte 3 vanno int=3)
... ... nulla (fine file) → EOF

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 91 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 93 di 100

es) C'è un file con una serie di interi uno per riga? Voglio metterli in un vettore e poi salvarli su un'altro file (meglio uno sullo stesso)

```
int main ()
{
  FILE * fp; int i=0;  FILE * fo; int nelem
  int v[100];                                  100 xché il max sono 100
  fp= fopen ("test.dat", "r");
  if (fp != NULL)
  { /* File ok, posso lavorarci */
     while (fscanf(fp, "%d" &v[i]) != EOF)
     { i++; }   nelem = i;
     fo= fopen ("his.his", "w");
     if (fo != NULL)
     { for(i=0; i<nelem; i++)
       {
          fprintf (fo, "%d\n", v[i]);
       }
     }
     else {errore}
  fclose (fp);                    l'ho aperto e poi devo/devo chiuderlo
  }
  else {errore}

return 0;
}
```

Osservazione

```
Fp = fopen ("_", "r");
if (fp == NULL)
{
   fprintf (stder, "errore");
   return -1;
```

Mi conviene usare if(fp == NULL) xché così intercetto immediatamente i casi sbagliati.

scrivo sullo schermo un messaggio di errore. Non uso return 0; xché si usa solo alla fine del main altrimenti torna all'inizio. Posso usare un qualunque numero. così indico real...

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 93 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 95 di 100

T.E sul robot) Un robot che si sposta a N, E, O, S di un certo valore (letto da un file) e voglio delle statistiche

```
#include <stdio.h>   #include <math.h>
#include <stdlib.h>
#include <ctype.h>
int punto_valido (float x, float y, float x1, float y2, float x2, float y2)
float distanza
int main (int argc, char *argv[])                    xcorr, ycorr
{ FILE *fp_mov, FILE *fp_coo;  float xorig, yorig; float x1,x2,y1,y2;
  /* argv[0] nome eseguibile                  float dist_lin=0.0, lung, char dir,
     argv[1] nome file con i movimenti          int invalidi=0, minvalidi=0,
     argv[2] nome "  "  coordinate della mappa    float tx, ty;
     argc==3
  */
  if(argc!=3)
  { fprintf (stderr, "Numero parametri errato. \n"
              "attesi: 2 parametri, ricevuti: %d\n", argc);
    return -1
  }
  fp_mov = fopen(argv[1], "r");
    if(fp_mov == NULL)
    { fprintf(stderr, "Errore", argv[1];
      return 2.}
  fp_coo=fopen(argv[2],"r");
        Ricontrollo se non ho dato errore
  fscanf (fp_coo, "%f%f%f%f", &x1, &x2, &y1, &y2);
  fclose (fp_coo);
  printf (" inserire le coordinate di partenza: ");
  scanf ("%f%f", &xorig, &yorig);
  if (! punto valido (xorig, yorig, x1, y1, x2, y2))
  { fprintf (stderr, "Il punto inserito è fuori dalla mappa!"); return 3;}
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 95 di 100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 97 di 100

```
printf("Spostamento max: %f\n", max);
printf("S...  ...h.in: %f\n", min);
printf("n° spostamenti validi: %d\n", nvalidi);
... "  "  "  "  invalidi  "  "  "  ", ninvalidi);
return 0;
}
float distanza
```

N.B Un file si memorizza quando:

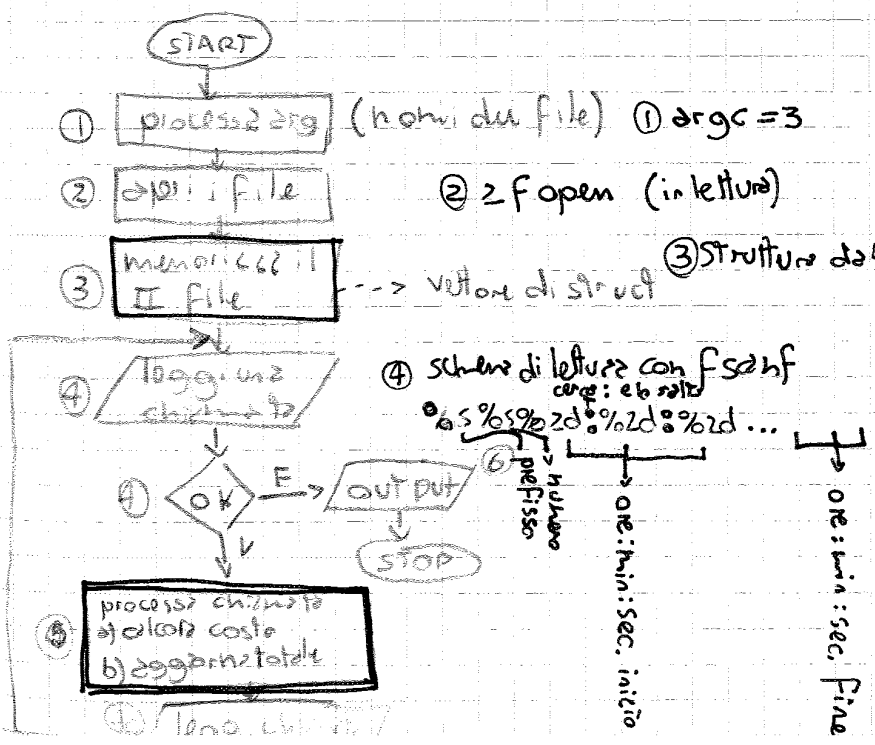
- è noto il n° di elementi (es. righe) del file
- il numero di elem. è ragionevole
- il file viene letto + volte e/o viene modificato

Un file non va MAI memorizzato quando:

- Non è noto il n° degli elementi
- il n° è noto ma è troppo grande
- il file è letto solo una volta e tipicamente non viene modificato.

ESERCIZIO 4) Vedi pc Vichi.

2) DDF alto livello



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 97 di 100