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: 1059 DATA: 02/09/2014

A P P U N T I

STUDENTE: Allora

MATERIA: Informatica

Prof. Laface-Acquaviva

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 96

DEBUG: fase in cui si tolgono gli errori (concettuali e di sintassi)

Dream Spark Dream msdn2.polito.it (Kiruto)

INFORMATICA

È la scienza che rappresenta (codifica) e manipola le informazioni

3 RIV. IND.

- MACCHINA A VAPORE '800
- ELETTRICITÀ '900 modo pulito e flessibile per trasportare energia

☺

B. Pascal: 1ª calcolatrice meccanica '600

Jacquard: automatizzazione della produzione tessile

Telescrivente 1801

Hollerith: schede perforate 1890

1° calcolatore elettronico II WW

↳ adesso non abbiamo + tubi a vuoto. Sono stati superati dai transistors (interruttori), poi dai circuiti integrati (migliaia di transistors), poi dai circuiti VLSI (VERY LARGE SCALE INTEGRATION)

PRIMA c'erano i NASTRI MAGNETICI (tipo VHS),  ACCESSO SEQUENZIALE

POI i NASTRI MAGNETICI  ACCESSO CASUALE tempo x k l'info arrivi: ½ giro

POI il DISCO MAGNETICO  ACCESSO CASUALE molto + veloce

↳ testina che fluttua (non tocca il disco)

i DISCHI ATTUALI hanno + DISCHI MAGNETICI  CD DVD X RAY → aumenta la densità di

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 3 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 5 di 96

INTELLIGENZA ARTIFICIALE:

- capacità di cercare in spazi amplissimi (es: simulatore di una partita di scacchi)
- capacità di inferire da un fatto qualcosa che non è nel fatto stesso

2012:

~~La scheda grafica del computer~~ è un SUPERCOMPUTER

architettura in cui gli elaboratori sono strettamente connessi e lo scambio di informazioni è velocissimo

IL PROBLEMA

↳ Ho dei dati e mi occorre la soluzione (che non è immediata).

Serve un processo costituito da una serie di passi logici.

Per es. Data l'area di un quadrato, ricavarne il perimetro.

Progettare

Difficoltà:

CERCARE UNA SOLUZ. "INFORMALE" FORMALIZZABILE

Algoritmo:

sequenza di operazioni che in un tempo finito permette di risolvere un problema.

Esempio:

PROBL: Calcolo del massimo tra A e B

SOLUZ: Il max è il + grande

SOLUZ. FORMALE:

1. max = 0
2. se A>B allora max=A; STOP
3. altrimenti max=B; STOP

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 5 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 7 di 96

CODEBLOCKS

↓
AMBIENTE INTEGRATO: ti fa tutto: compilare, debug, stesura, ecc.

Creo un nuovo progetto [CONSOLE APPLICATION] in C

Uso GNU GCC Compiler

Normalmente conviene fare solo la configurazione in debug

Sul Management vado su main.c in Sources (↳ editing)

```
1  #include <stdio.h>     ] PSEUDOISTRUZ. X IL COMPILATORE
2  #include <stdlib.h>    ]
3
4  int main()
5  {
6      printf("Hello world!\n");
7      return 0;
8  }
```

- `<stdio.h>`: header, intestazione; file intestazione → prima di leggere il file vai allo stdio per capire le mie istruzioni
- `<stdlib.h>` → per il return
- `\n` dice "vai a capo"
- `;` → fine di un'istruzione
- `"..."` → STRINGA di caratteri
- `return 0;` → deve sempre esserci uno zero
- `{ }` → corpo del programma

Vado poi su "build" sulla barra in alto.

USO il DEBUGGER, la finestra nera che si apre è la console.

Con "next line" correggo i "break point", punti dove la lettura del progr. si interrompe.

Nel mio esempio:

```
1  #...
2  #...
3  #include <math.h>
3
4  int
5  main (){
6  float area, lato, perimetro;
7      printf ("Inserire area\n");
       scanf ("%f", &area);
       lato = sqrt(area);
       perimetro = lato*4;
       printf ("Perimetro = %f\n", perimetro);
       return 0;
   }
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 7 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 9 di 96

FILE SORGENTE → COMPILATORE → FILE OGGETTO → LINKER ← LIBRERIE → FILE ESEGUIBILE → può essere direttamente caricato in memoria ed eseguito

IDE = Ambiente integrato

INTEGRATED DEVELOPMENT ENVIRONMENT

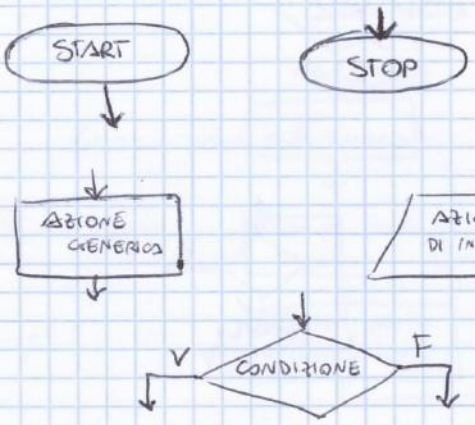
DIAGRAMMA DI FLUSSO

PROGRAMMARE: scrivere un documento (file sorgente) che descrive il problema in oggetto

STRUMENTO GRAFICO K RAPPRESENTA L'EVOLUZIONE LOGICA DELLA RISOLUZ. DEL PROBLEMA,

è composto da:
- blocchi elementari per descrivere azioni/decisioni (binarie)
- archi orientati per descrivere la sequenza dei blocchi

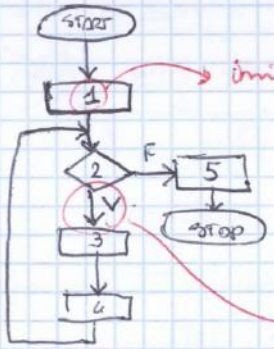
Blocchi



Console: immettere dati e visualizzare i risultati

Blocco di connessione

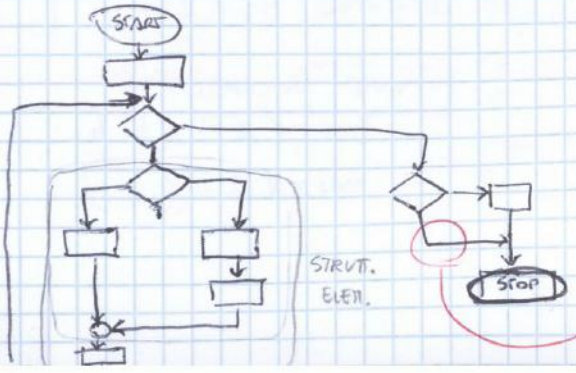
Es:



inizializzazione

→ while 2 è vera, continua a ciclare

Altro es:



in questo caso non si faccia niente.

7

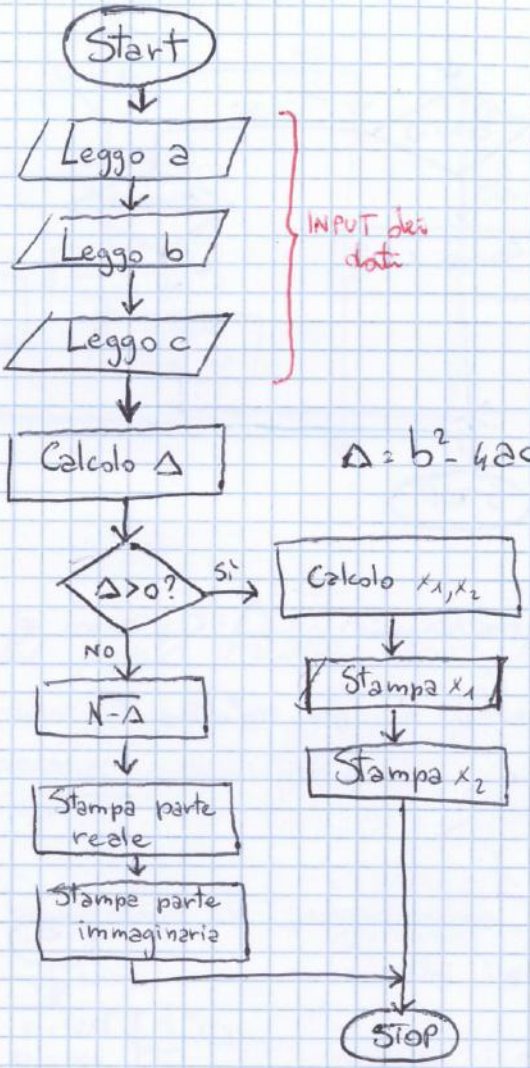
© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 9 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 11 di 96

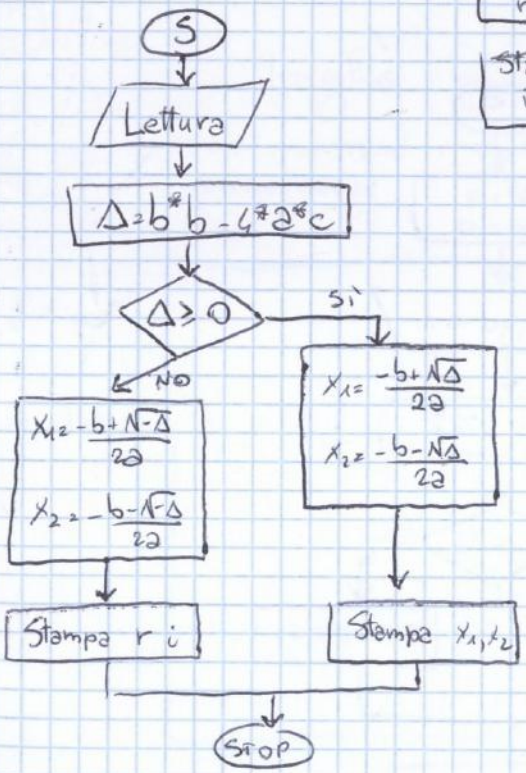
MEMORIA: occorre che il computer memorizzi di volta in volta il TOT di numeri pari/dispari

2) $ax^2+bx+c=0$

Dati a, b, c



IN FORMA MIGLIORE:



Se a=0 il programma "ABORTISCE" e avvisa che c'è questo tipo di errore.

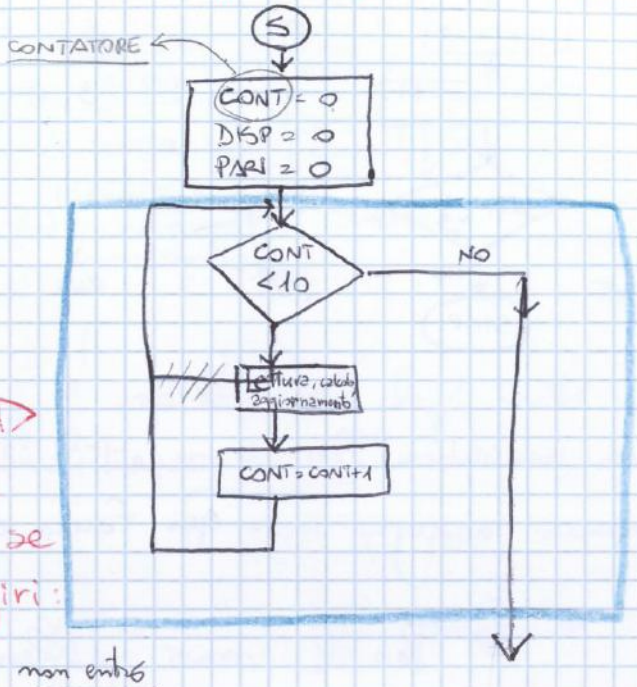
Devo sempre tenere conto dei valori strani!

Questo algoritmo è SBAGLIATO.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 11 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 13 di 96

Userò un ciclo per ripetere il blocco



Blocco strutturato While

Trucco per controllare se faccio 10 giri:

0 numeri: non entro

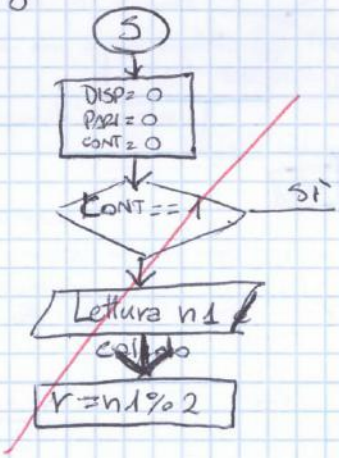
1 numero: 1 giro

Avrei anche potuto mettere CONT = 1 e 

PERÒ è meglio usare CONT = 0

Altro es:

Voglio far girare il programma finché non incontra un numero negativo.

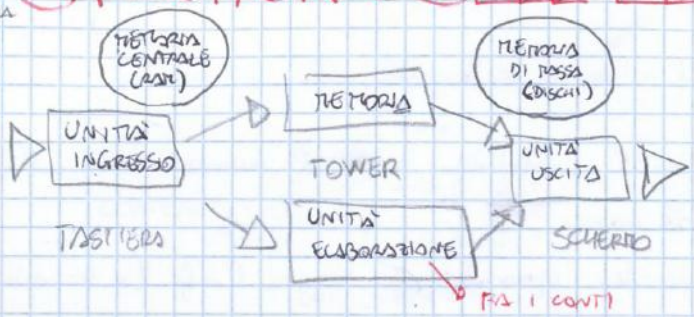


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 13 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 15 di 96

STRUTTURA DELL'ELABORATORE

ARCHITETTURA



RAM (RANDOM ACCESS MEMORY) = il tempo che ci metto per arrivare all'indirizzo 0 all'indirizzo 100.000.000 è lo stesso

[non procedo sequenzialmente]

è VOLATILE, le info si perdono quando la macchina viene spenta

Per questo mi serve il DISCO

I CHIP FONDAMENTALI

SINONIMO DI MICROPROCESSORE

Microprocessore (μP) → Realizza le funzioni di una CPU (Central Processing Unit)

Ha 3 cose al suo interno:

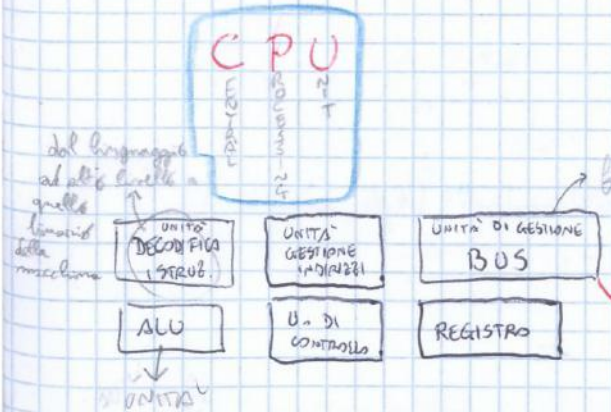
1. UNITÀ ARITMETICA & LOGICA: +, −, ecc. confronti (ex. CONT<16) — ~1 NANOSECONDI
2. INSIEME LIMITATO DI MEMORIA (da una decina a max 128 REGISTRI di memoria) → DIMENSIONE DI 1 WORD (max 64 bit)

Libro Papy pg. 6 / 40

La 1ª unità lavora SUI REGISTRI

3. UNITÀ DI CONTROLLO: compito di andare al prossimo dato o istruzione (è il cervello del μP)

~60 ns NANOSECONDI



Memoria centrale - RAM

CARATTERISTICHE

- TRASPORTANO 1 DATO X VOLTA
- Freq = n DATI TRASPORTATI / sec
- Ampiezza = n. di bit da cui è formato un dato
- SE TSL DIMENSIONATO → COLLO DI BOTTIGLIA

fili che collegano (n SISTEMA CIRCOLATORIO) in cui transitano info

3 TIPI

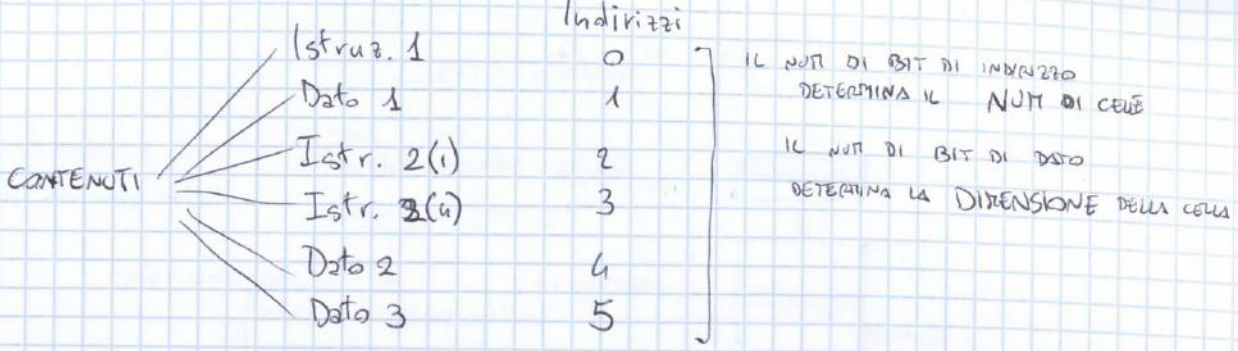
SEGUE

13

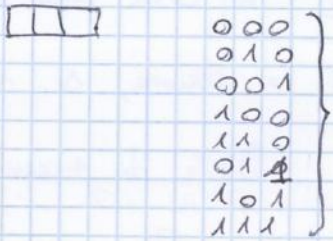
© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 15 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 17 di 96

Organizzaz.



Immaginiamo di avere 3 fili in un abus BUS.



$2^3 = 8$ qste sono tutte e sole le combinaz. possibili.

$\Rightarrow$ Ho solo 8 indirizzi.

Devo avere molti più fili.

Dati N fili, ho 2^N indirizzi

DBUS:

Con 8 fili Posso leggere in solo byte (8 bit) per volta

64 fili $\Rightarrow$ 8 byte in una volta

Nei primi computer della Microsoft c'erano 20 fili

$2^{20} =$?

RICORDA:

$2^5 = 32$

$2^8 = 256$

$2^{10} = 1024$ 1K

$2^{20} =$ 1M

$2^{30} =$ 1G

$2^{40} =$ 1T

32 fili

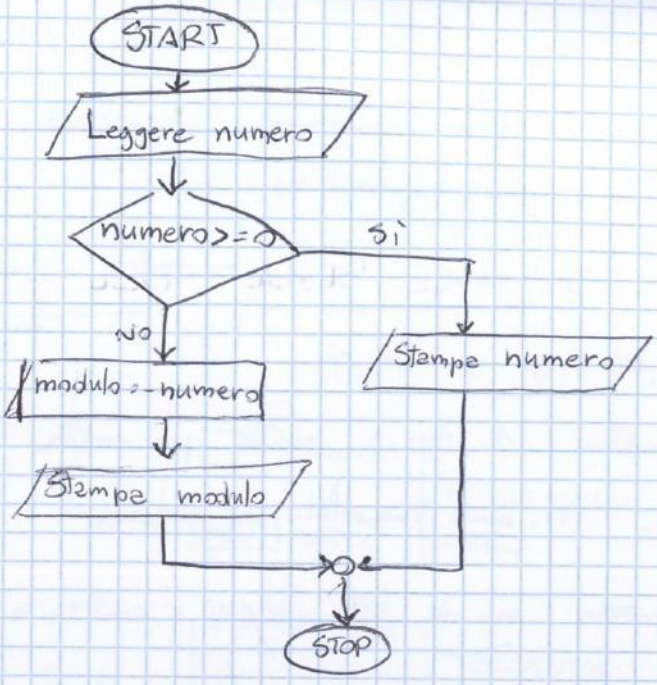
$2^{32} = 2^{30} \cdot 2^2 = 4G$

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 17 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 19 di 96

Programma:

Scrivere il modulo di un numero



ES. DI ISTRUZIONI ASSEMBLER

nel microprocessore	MOVE A, dato 1	3E	20
	MOVE B, dato 2	3F	03
	add A,B (1 byte)	C6	
	store risul, A (memorizza il risultato in A)	10	20 H

LINGUAGGIO ASSEMBLER

LINGUAGGIO MACCHINA

Formato delle istruzioni

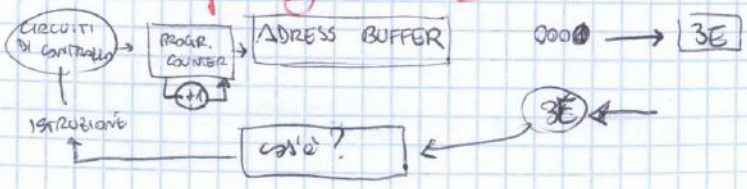
COD. OPERATIVO — OPERANDI

campo sempre presente, indica l'operazione che la CPU (CU+ALU) deve eseguire

Linguaggio macchina

3E
20
27
⋮

Esecuz. programma



clock (freq.) — n operaz. / secondo

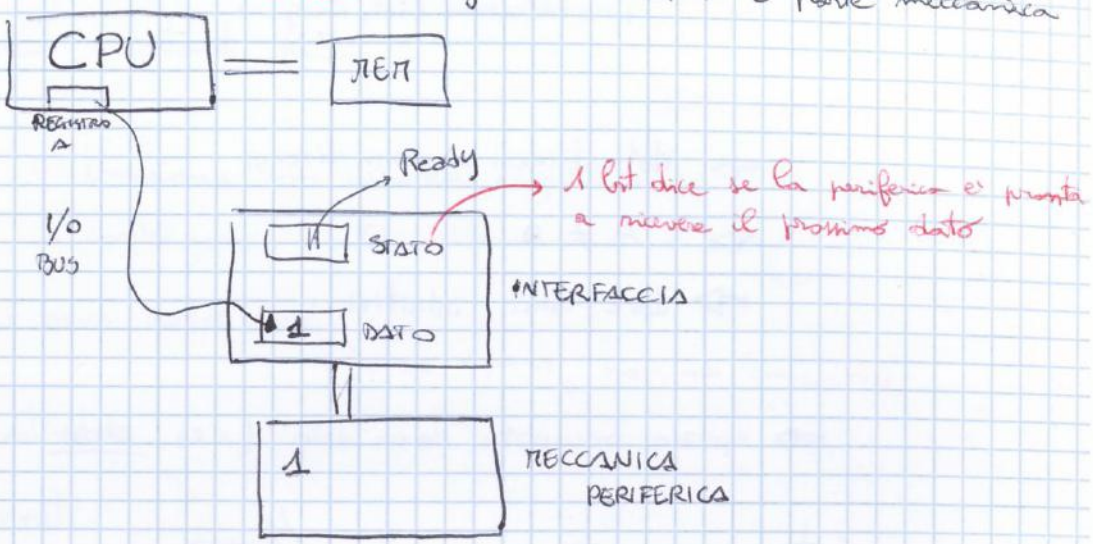
© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 19 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 21 di 96

I/O

Trasforma info dal mondo umano a quello del computer

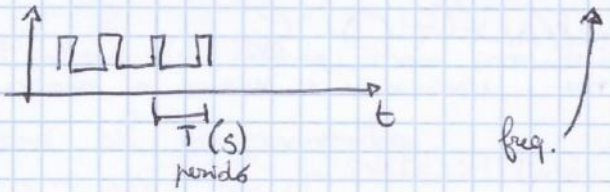
Interfaccia: collega parte elettrica e parte meccanica



CLOCK

elemento di temporizzazione presente in ogni elaboratore, genera un riferimento temporale comune per tutti gli elementi costituenti il SISTEMA DI ELABORAZIONE.

Hertz: impulsi al secondo



CICLO MACCHINA: multiplo di T, tempo impiegato per compiere una operazione elementare

UN'ISTRUZ. richiede più cicli macchina

2ª SETTIMANA

LINGUAGGI AD ALTO LIVELLO

ELEMENTI:

- PAROLE CHIAVE (Keyword)
- DATI
- IDENTIFICATORI (variabili)
- ISTRUZIONI

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 21 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 23 di 96

- FILE
- ETICHETTE

Sono i NOMI che uso nel programma per individuare costanti, file, ...

Regole:

- iniziano con carattere alfabetico o "_" (per es. 3X non va bene)
- contengono caratteri alfanumerici o "_" (per es. X3 va bene)

NON DEVO AVERE SPAZI:

"ciao mamma" non va bene.

Caratteristiche

- Riservati:
  - PAROLE CHIAVE DEL LINGUAGGIO
  - ELEMENTI DELLA LIBRERIA C STANDARD

  } NON POSSO CHIAMARE UNA VARIABILE printf

- Interni: le entità del programma
  - CASE SENSITIVE (A ≠ a)
  - Significativi almeno 31 caratteri

Commenti

Testi liberi all'interno del sistema non considerati dal compilatore, che servono al programmatore.

FORMATO:

- /* */ POSSO ANCHE METTERLO SU + LINEE

  es. /* ciao mamma
  sono a Torino */

- // FINO A FINE LINEA

  es. // ciao

- NON POSSO METTERE UN COMMENTO DENTRO L'ALTRO (ANNIDARLI)

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 23 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 25 di 96

Esempio:

```
#include <stdio.h>
int main(void)
{
    return 0;
}
```

= main(), ma è formalmente + corretto

int perché se le cose vanno bene main() alla fine restituisce il risultato 0, che è intero

= vuoto, non ha parametri

I DATI NUMERICI

DICHIARAZIONE DATI:

```
int x;
```

TIPO — variabile

- allocar.
- assegnat.

Il calcolatore va nella RAM e "alloca" lo spazio per 32 bit e gli "assegna" un nome

Specificare la MODALITÀ DI ACCESSO

per es. `const int x = 3;` → devo dare un valore, perché dopo nel programma non posso farlo.

GUAI

SE CAMBIO IL VALORE DI X NEL PROGR.

Posso però fare

```
main(){
int y=7;   → dichiarazione di variabile
:
y = 10;
```

TIPI BASE

char	intero a 8 bit	$2^8 = 256$ combinazioni → caratteri: A,B,...,Z, 0,...,9, a,b,...,z, ;,.,/,... (A=65, 0=48, a=97) 0,...,255
int	interi (complemento a 2)	32 bit
float	reali (floating point singola precisione)	
double	reali (" " doppia ")	

PRIMA LEGGO IL NUMERO, POI LO POSSO CONFIGURARE CON UN CONTENERE

%od sta per "digit"

ho così codificato i caratteri.

posso anche mettercì un numero

Per es. `char num = 3;` — num, non Keyboard

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 25 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 27 di 96

VARIABILI

Locazioni di memoria destinate alla memorizzaz. di dati dal valore modificabile.

Posso fare delle liste di variabili

```
int x1, x2, x3
```

Es. di assegnazione:

```
Float a,b;
a = 3.1;
b = 6,02e23;
```

inizializzazione

b = 6,02 · 10^23

incremento

```
a = a+1;
```

COSTANTI

```
const <tipo> <costante> = <valore>
```

Es:

```
const double PIGRECO = 3.141593;
const char SEPARATORE = '$';
```

I SINGOLI APICI trasformano il numero nel CARATTERE

In genere gli identificatori di costanti sono MAIUSCOLI

Costanti di tipo int, short, long

- 26
- 0x12 → numero esadecimale
- 26L (long)
- 26u (unsigned)
- 26UL

float, double

- 216.3

COSTANTI SPECIALI

4 numeri tra i 256 di cui (?)

'\n'	line feed
'\t'	tab
'\b'	backspace
'\f'	form feed

VISIBILITÀ VARIABILI

Scope: area di validità delle variabili

- Var. globali → visibili al main e in tutti i sottoprogrammi

- Var. locali → le variab. non hanno senso fuori dal main()
  ex. la printf contiene un progr. con variabili locali che noi non

25

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 27 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 29 di 96

• RAPPRESENTAZ. DI N CIFRE

$$A = \sum_{i=0}^{N-1} a_i \cdot B^i$$

Es: $101_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 5_{10}$

↳ SISTEMA BINARIO

x es. 4 bit

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8

2^4 numeri $= 16$ [0-15]

DAL SIST. DECIMALE A QUELLO BINARIO:

Es: 13_{10}

13	2	
6	2	
3	2	
1	2	
0		

resti: 1, 0, 1, 1

1101_2

⇒ I NUM PARI FINISCONO PER 0
I NUM DISPARI " " 1

TERMINOLOGIA

1 0 1 1 0 1 1 0

Most Significant Bit

Least Significant Bit

PESO CHE MOLTIPLICA L'INDICE + ALTO

LIMITI DEL BINARIO

• NUM. NATURALI IN BINARIO

1 bit ~ 2 numeri ~ $\{0,1\}_2 = [0 \dots 1]_{10}$

N bit ~ 2^N numeri

$0 \le x \le 2^N - 1$ [base 10]

$(000\dots0) \le x \le (111\dots1)$ [base 2]

27

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 29 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 31 di 96

Ultimo es.

$$\begin{array}{r} 1111 - \\ 0011 = \\ \hline 1100 \end{array} \qquad \begin{array}{r} 15 - \\ 3 = \\ \hline 12 \end{array}$$

$1100 = 2^3 + 2^2 =$ 12

COSÌ FUNZIONA

Il problema del primo es. è che il risultato, che è negativo, NON È RAPPRESENTABILE, visto che il range va da

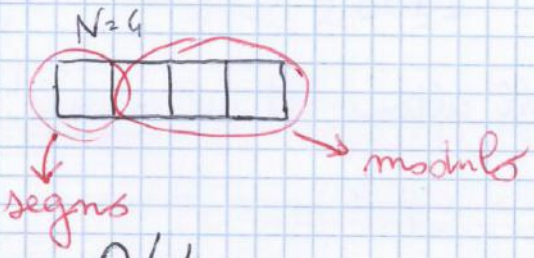
$0 - 15 \quad (= 0 - 2^4 - 1)$

UNDERFLOW

UNDERFLOW ↓ sottrae. | $0 \dots 2^4 - 1$ | OVERFLOW ↓ somma

NUMERI CON SEGNO

RAPPRESENTAZ. MODULO E SEGNO:



0/1
\+ −

Per es. $-3 = 1011$ (1 = −, 011 = 3)

PERÒ spreco un bit per il segno!

VALORE + PICCOLO: 1111 = −7

VALORE + GRANDE: 0111 = +7

Ho 2 ZERI: 1000 / 0000 → ho sprecato una combinazione

CON N bit

$$-2^{N-1}+1 \quad \dots\dots \quad 2^{N-1}-1$$

2 DIFETTI:

- IL DOPPIO ZERO
- IL COME SI FANNO SOMME E SOTTRAZIONI

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 31 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 33 di 96

- Se MSB=0 ho il sistema binario normale [DETTO NATURALE]

Es: 0011

$-2^3 \times 0 + 2^2 \times 0 + 2^1 \times 1 + 2^0 \times 1 = +3$

→ 1ª analogia con MODULO & SEGNO

- Se l'MSB=1 il numero è senz'altro negativo

Es: 1011

$-2^3 + 2^1 + 2^0 = -5$

→ 2ª analogia con MODULO & SEGNO

viceversa il num è POSITIVO

NUM. NEGATIVO + PICCOLO:

1000

-2^3

$= -8$

Risp. a modulo e segno ho 1 numero in +!

$-2^{N-1} \dots 2^{N-1} - 1$

NUM POSITIVO + GRANDE:

0111

$2^2+2^1+2^0$

$= 7$

Somme e sottrazioni

SI FANNO DIRETTAMENTE, senza badare ai segni degli operandi

Se ho un num. binario, come rappresentarlo in CA2?

Es: PRENDO IL MODULO:

0011 = |-3|

INVERTO I BIT 1100 + (1→0, 0→1)

SOMMO 1 1 =

1101

CONTROLLO: $-2^3+2^2+2^0 = -8+4+1 = -3$

SOMMA IN CA2

00100110 + 32+4+2 = 38 +

11001011 = −128+64+8+2+1 = −53 =

11110001 −15

$-128+64+32+16+1 =$

PERÒ

COME VEDO L'OVERFLOW?

Qui non è determinato dal carry sull'ultimo bit

31

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 33 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 35 di 96

IEEE 754 SP

4 byte

segno	esponente	mantissa
1 bit	8 bit	23 bit

può essere negativo ed è in CA2

modulo e segno

IEEE 754 DP

8 byte

segno	esponente	mantissa
1	11	52

Esempi di notaz. scientifica

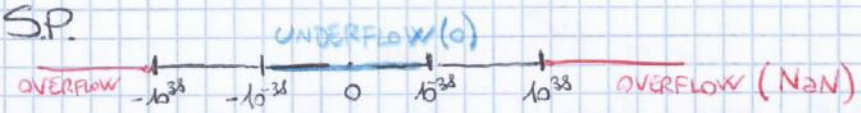
$3,14 = 0,314 \cdot 10$

$0,0001 = 0,1 \cdot 10^{-3}$

$137 = 0,137 \cdot 10^{3}$

RISOLTO IL PROBLEMA DELLA VIRGOLA

S.P.



NOTAZ. ESPONENZIALE ⟹ intervalli non regolari tra 1 num. e il successivo. [vicino allo zero i num. sono + fitti]

Visto il diverso modo di considerare i bit serve uno strumento di calcolo apposito, l'FPU (Floating Point Unit)

Il cellulare non ce l'ha, ma la emula in software (molto + lentamente)

Finiamo così di rappresentare i numeri

Qualsiasi insieme numerabile può essere rappresentato come sequenza di numeri [per es. i caratteri dell'alfabeto]

CARATTERI

codifica standard: - ASCII
- EBCDIC

ASCII: 8 bit

52 caratteri alfabetici (a...z/A...Z)

10 cifre

segni di interpunzione (.,;:...)

caratteri di controllo ~~(tab)~~ (NL [= new line], ...)

UNICODE: esprime tutti i caratteri esistenti

UTF-8 è la codifica di Unicode + usata

↳ estensione dell'ASCII

1 byte ASCII
⋮
4 byte caratteri rarissimi

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 35 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 37 di 96

SALTA LA PAG-

$x-2 \quad x+1$

$$\frac{-1}{\frac{x^2-4x+4+x^2-2x+1}{(x-2)^2}\,(x-2)^2}$$

1 presentaz. numeri
2 architettura
3 operatori - form. logica, tabelle di verità.

$2xe^{2x} = o\left(\frac{1}{x^2}\right)$

$2x^2-6x+5$

$\frac{\Delta}{4} = 9-10$

5 febb 13:00
7D
Risultato entro lunedì

$$\lim_{x\to-\infty} \frac{2xe^{2x}}{\frac{1}{x^2}} =$$

$-x+2-1+x$

$$\lim_{x\to-\infty} 2x^3e^{2x} =$$

$2-6+5 = -1$

$$= \lim_{x\to-\infty} \frac{2x^3}{\left(\frac{1}{e^{2x}}\right)}$$

$$\frac{1}{x^2-4x+4+1-2x+x^2}$$

$\frac{0}{\infty}$

$\frac{6x^2}{-2e^{-2x}}$ $\frac{12x}{-4e^{-2x}}$ $\frac{12}{-8e^{-2x}}$ $2x^2-6x+5$

$= -\frac{12}{8}e^{2x} = 0$

$$\frac{\frac{1}{x^2}}{\frac{1}{e^{\frac{1}{x}}}} \quad \infty \quad \infty = \frac{\frac{1}{x^2}}{e^{-\frac{1}{x}}} = \frac{12\frac{1}{x^3}}{+\frac{1}{x^2}e^{-\frac{1}{x}}}$$

$$\frac{2\cos 2x}{} \quad \frac{+2\cos 2x + 2x - 2\sin 2x + 4x}{x^3}$$

$$\frac{\frac{1}{x}}{e^{-\frac{1}{x}}} = \frac{-\frac{1}{x^2}}{-\frac{1}{x^2}e^{-\frac{1}{x}}} = \frac{1}{+\infty} =$$

$e^t = 1+$

$$\frac{2\cos 2x}{} \quad 2x\left(\frac{1+\cos 2x}{x^3}\right) - \frac{2\sin 2x}{x^3}$$

$\frac{e^{\frac{1}{x}}}{x^2} \le \frac{1}{x^2}$ $\quad \frac{e^{\frac{1}{x}}}{x^2} \sim \frac{1+\frac{1}{x}+\frac{1}{x^2}}{x^2} = \frac{x^2+x+1}{x^2}$

$0 < \frac{e^{\frac{1}{x}}}{x^2} \le \frac{1}{x^2}$

$$\frac{-2\cos 2x + 2 - 4x\sin 2x}{3x^2}$$

$\times \log x$

$$\frac{e^{\frac{1}{x}}}{\frac{1}{x^2}} \qquad \frac{\log x}{\frac{1}{x}}$$

$$\frac{2\cos 2x + 4x\sin 2x + 2 - 4\cos 2x}{3x^2} =$$

35

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 37 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 39 di 96

I/O FORMATTATO AVANZATO

printf

- giustificare la stampa a sx
- mette sempre il segno

% [flag] [min dim] [. precisione] [dimensione] <carattere>

dimensione minima di stampa caratteri

Es:

%.3f ⟹ 5.000

%5d ⟹ ____6 (4 caratteri vuoti) serve per allineare, tipo

____3

___12

%+0flagd ⟹ +37

scanf

% [*] [max dim] [dimensione] <carattere>

* fa saltare un argomento

%*c vuol dire "salta il carattere"

PREPROCESSORE C (è un programma che lavora prima del compilatore)

Attua un passo preliminare che prevede la vera e propria traduzione

La sua funzione principale è espandere le direttive che iniziano con '#'

DIRETTIVE PRINCIPALI:

1) #include <stdio.h> → libreria di standard I/O che contiene printf e scanf

direttiva di preprocessore: il preprocessore cerca nell'hd dischi il file di testo stdio.h e lo copia, mettendolo al posto di #...

2) #define <COSTANTE> <valore>

identificatore della costante simbolica

tutti i PIGRECO sono sostituiti da 3.14

Es: #define PIGRECO 3.14

Il file è + leggibile e flessibile [cambio il valore una sola volta per tutte]

→ CAPISCO COSA STA FACENDO

E' diverso da es. const float PI = 3.14 ⟹ qst occupa uno spazio in memoria, l'altro modo no.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 39 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 41 di 96

Le espressioni possono essere messe all'interno della printf

Per es. printf("A div B = %d \n", a/b);

OPERATORI DI CONFRONTO:

UGUAGLIANZA
- UGUALE `a == b`
- DIVERSO `a != b` → NB

ORDINE
- MAJOR `a > b`
- MINOR `a < b`
- MAJOR O UGUALE `a >= b`
- MINOR O UGUALE `a <= b`

ASSEGNAZIONI CON OPERATORI DI INCREMENTO

`a = a+1;` ⟺ `a++`

`a = a-1;` ⟺ `a--`

Per fare `a = a+2;` ⟺ `a+=2`

NOTAZIONE:
- POSTFISSA `x++;`
- PREFISSA `++x;`

VARIABILE MODIFICATA PRIMA DELL'OPERAZIONE

es: x = 4

`y = x++` POSTFISSA → x = 5, y = 4

`y = ++x` PREFISSA → x = 5, y = 5

sempre

OPERATORI RELAZIONALI

(POSSONO ESSERE CHIESTI CONE TEORIA)

- `<` `>` `>=` `<=` `!=` `==` → risultato booleano
- risultato sempre int
  - risultato = 0 FALSO
  - " ≠0 VERO

AND &&

OR ||

NOT !

REGOLE DI PRECEDENZA:
NOT > AND > OR

es: (!A) && B

(x cambiare l'ordine basta usare le parentesi)

es: !(A && B)

USARE SEMPRE LE PARENTESI!

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 41 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 43 di 96

NOT's table

A	$\bar{A}$
F	V
V	F

AND && · ×

OR || + → INFATTI È ≠0 SE ALM. 1 VARIABILE È ≠0

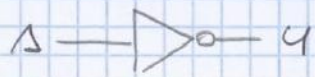
NOT ! A' $\bar{A}$ ~A

INFATTI SE C'È UN F (cioè uno 0) VALE ZERO

I bit sono 0/1 ⇒ VARIABILI BOOLEANE

I circuiti di base del calcolatore si fanno con delle porte logiche

- **PORTA INV / NOT** → INVERTER



$Y = A'$

- **PORTA AND**

$Y = A \cdot B$

- **PORTA OR**

$Y = A + B$

Altro operatore importante:

XOR ⊕ Å → PERCHÈ ASSOMIGLIA A "OR"

vero solo se $A \neq B$ ⇒ **Rileva le differenze**

A	B	A Å B
V	V	F
V	F	V
F	V	V
F	F	F

- **PORTA XOR**

PROVVISORIO: $Y = A \oplus B = A \cdot B' + A' \cdot B$

A	B	XOR	A·B' + A'·B
0	0	0	0 + 0 → 0
0	1	1	0 + 1 → 1
1	0	1	1 + 0 → 1
0	0	0	0 + 0 → 0

ORDINE BINARIO NATURALE CRESCENTE

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 43 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 45 di 96

RANGO ESPRESSIONI ARITMETICHE

Es: intero / intero = RISULTATO TRONCATO

char
short
unsigned short
int
unsigned int
long
unsigned long
long long
unsigned long long
float
double
long double

NELLE OPERAZIONI "VINCE" IL RANGO ~~GRADO~~ + ALTO [si eleva il rango dell'espressione]

int/double = double

QST VALE ANCHE PER LE COSTANTI NUMERICHE

int/3 = intero
int/3.0 = double

Però PER EVITARE DI CHIAMARE fare la double un' int solo per la divisione

CAST

c = (double) a/b

a diventa MOMENTANEAMENTE un double

CAMBIO TEMPORANEAMENTE IL TIPO DI UNA VARIABILE [COSÌ CAMBIA IL TIPO DELL'ESPRESSIONE]

OPERATORE sizeof()

Anche questo lavora sui tipi e non sulle variabili.
Restituisce la DIMENSIONE IN BYTE

x = sizeof(float) <=> x = 4

Serve per verificare le dimensioni di un certo tipo in una certa macchina.

IF: SCELTE ANNIDATE

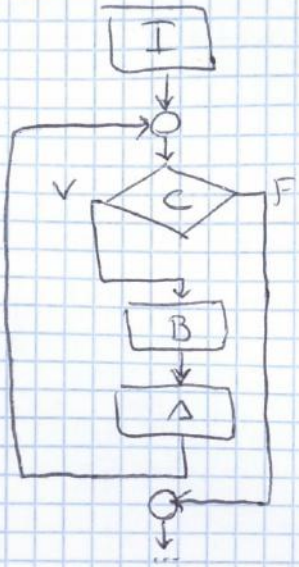
```
if(c1) {
    A1;
    if(c2){
        A2;
    }
} else {
    A3;
}
A4;
else {
    B;
}
```

INDENTAZIONE

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 45 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 47 di 96

<comp6>
B
}



Nelle graffe ho solo il comp6.

Es1:
dati in input carattere ch e N intero, stampare N volte ch

```
For (i=0; i<N, i++)
    printf("%c", ch);
printf("\n");
```

↔

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

lo metto fuori dal ciclo per dare un "a capo" finale

→ forse nel ciclo, avrei dei ~~risultati~~ caratteri incolonnati.

Es2:
INTRODURRE 100 NUMERI INT E CALCOLARNE LA MEDIA. SI CONTROLLI CHE OGNI NUMERO È $0 \le x \le 30$; ALTRIMENTI IL RISULTATO VA IGNORATO.

```
#include <stdio.h>
main(){
int valore, i, Totale=0, M=0;
    const int N=100;
    For (i=0; i<N; i++)                    /* PER OGNI VALORE INTRODOTTO */
    {
        scanf("%d", &valore);              → leggo l'input e poi lo consumo, non mi serve a niente altro dopo che l'ho sommato a Totale. La variabile può essere riutilizzata.
        if (valore<0 || valore>30)         /* CONTROLLO VALIDITÀ */
            printf("Valore non valido");
        else
        {/* caso normale */
            Totale += valore;              /* ACCUMULA IL NUOVO VALORE IN Totale */
            M++;                           /* HO LETTO UN DATO IN PIÙ */
        }
    }
    CONTINUA
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 47 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 49 di 96

Esempio:

```
i=0;
while (i<N)
{
  j=0
  while (j<N)
  {
    printf("%d, %d", i, j);
    j=j+1;
  }
  i=i+1;
}
```

N=5

0,0
0,1
0,2
0,3
0,4
0,5
1,0
.
.
.
ecc.

25 giri

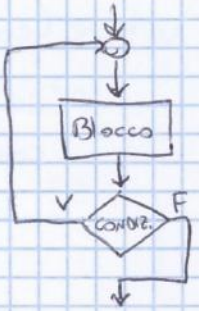
ISTRUZIONE do

```
do <blocco>
while (<condizione>);
```

→ se ha + di 1 istruzione, uso le graffe

→ condiz. booleana

CICLO REPEAT,
eseguito almeno 1 volta



Es: Leggere N e controllare N≥0, altrimenti rileggere.

vd slides

```
do
  scanf("%d", &n);
while (n<=0);
```

⟷

```
scanf("%d", &n);
while (n<=0)
  scanf("%d", &n);
```

INTERRUZIONE DEI CICLI

[x MODIFICARE IL NORMALE FLUSSO DI UN CICLO]

- **break:**
  Termina il ciclo
  ⇒ va all'istruz. fuori dal ciclo

- **continue:**
  Termina l'iterazione corrente
  ⇒ va alla prox iterazione del ciclo

47

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 49 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 51 di 96

Senza break: L STRUTTURATA!

```
int valore, finito=0
while(scanf("%d", &valore) && !finito)
{
  if (valore==0)
  { printf("Valore non consentito\n");
    finito=1;   ⟹  !finito = 0
  }
  else
  {
    /* altre istruz. del ciclo */
  }
```

variabile usata come un flag:

FLAG

Variabile in uso booleano per bloccare i cicli (di solito).

Con il continue:

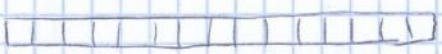
```
while(scanf("%d", &valore))
{
  if (valore==0)
  {
    printf("Valore non consentito");
    continue;
  }
  printf("Istruz. dopo il continue\n");
}
```

→ QUI LO ZERO NON INTERROMPE IL CICLO, VIENE IGNORATO

VETTORI o arrays

QUANDO HO TANTISSIMI NUMERI, ANZICHÈ USARE TANTISSIME VARIABILI, USO UN VETTORE IN CUI METTERLI

a



USO L'INDICE per scorrere il vettore e trovare l'entrata n-esima

variabile che varia da 0 a TANTISSIMO(= NUM DI NUMERI)

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 51 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 53 di 96

L'INDICE deve essere INTERO (anche un'espressione intera)

Es:
```
double a[100];
int x,y,z,a;
x = a[ i+32-j ];
```

Vettori e cicli [soprattutto "for"]

```
int data[10];
for (i=0; i<10; i++)
{
    //operazione su data[i]
}
...
```

La DIMENSIONE del vettore deve essere una COSTANTE

GUAI se è una variabile!
↳ anche se la variabile non cambia.

non userò mai un vettore se la dimensione non è nota a priori.

NON POSSO NEMMENO TIRARE A CASO UNA DIMENSIONE MOLTO GRANDE

COPIA DI UN VETTORE

NON POSSO FARE ~~W=V~~

Copio con un ciclo for ogni elemento di V in W

```
for(i=0; i<N; i++)
{
    W[i] = V[i];
}
```

Es: LEGGERE DA TASTIERA 10 VALORI, METTERLI IN UN VETTORE E CALCOLARE min e MAX.

[NON SAREBBE NECESSARIO UN VETTORE]

51

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 53 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 55 di 96

```
.else{
    while(num !=0){
        v[i] = (num % 2);   /* resto della div. per 2 */
        num = num / 2       /* div. intera per 2 */
        i--;
} }
for (i=0; i<10; i++)
    printf ("%d", v[i]);
printf ("\n");
}
}
```

Ricerca di un elemento

DATO UN VALORE NUMERICO, VERIFICARE

- ALMENO UN elemento ha quel valore
- SE SI' DIRE DOV'È
- SE NO, DIRE CHE NON ESISTE

PROBLEMA DI "ricerca di esistenza"

Es:

```
int dato;     /* dato da ricercare */
int trovato;  /* Flag x la ricerca */
int pos;      /* posizione di un elemento */
...
printf ("Elemento da ricercare?");
scanf ("%d", &dato);
trovato = 0;
pos = -1;
for (i=0; i<N; i++)
{
    if (v[i] == dato)
    {
        trovato = 1;
        pos = i;
    }
}
if (trovato == 1)
    printf ("Elemento trovato alla posiz. %d", pos);
else
    printf ("Elemento non trovato");
...
```

MEMORIZZA BENE IL MECCANISMO

/* per memorizzare solo il primo elemento in cui c'è dato, posso scrivere for(i=0;(i<N)&&(trovato==0); i++) */

/* memorizzo dov'è l'elemento di interesse */

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 55 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 57 di 96

```
    if (duplicato == 1)
        printf("V[%d] e' duplicato\n", i);
}
```

In questo algoritmo però la stessa coppia di doppioni viene contata 2 volte. Per evitarlo nel secondo ciclo si può porre come inizializzazione $j = i$

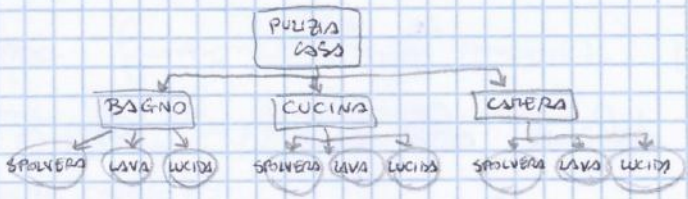
SOTTOPROGRAMMI

UN PROGRAMMA NORMALE HA MIGLIAIA DI ISTRUZIONI

Approccio top-down

- DECOMPONGO IL PROBLEMA IN SOTTOPROBLEMI + SEMPLICI, FINO A PROBLEMI ELEMENTARI
- RIPETIBILE SO + LIVELLI

Es:



Spolvera, lava e lucida sono 3 funzioni, programmi che uso più volte nel main()

Anche main() è una funzione

PROCEDURE: SOTTOPROGRAMMI CHE NON RIPORTANO UN RISULTATO ex. la printf

FUNZIONI: RITORNANO UN RISULTATO (PRIMITIVO O NON) ex. math.h dà la radice di un numero

Procedure e funzioni hanno dei PARAMETRI (o ARGOMENTI)

parametri → f ⇒ risultato (intero come un'operazione)

Nel C K&R esistono solo funzioni, anche se magari ignoro il risultato;

Dal C89(ANSI) in poi (void) impone che il ritorno venga ignorato ⇔ procedura

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 57 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 59 di 96

PARAMETRI:

Così come il risultato, sono sempre associati a un TIPO

Es: float media(int a, int b)

int a
int b ⇨ media() ⇨ float

N.B. TIPI vanno poi rispettati

Es:
```
float x;
int a,b;
x = media(a,b);
```

L'USO DELLA FUNZIONE È NELLA FORMA DI UNA NORMALE ISTRUZIONE.

Una funzione (non main()!) può anche invocare se stessa, come se si duplicasse.

Es:
```
# include <stdio.h>
int modabs(int v1, int v2);      // prototipo
main()
{
    int x,y,d;
    scanf("%d %d", &x, &y);
    d = modabs(x,y);             // utilizzo (anche con variabili diverse dei parametri!)
    printf("%d\n", d);
}
int modabs(int v1, int v2);      // definizione
{
    int v;
    if (v1>=v2)
        v = v1-v2;
    else
        v = v2-v1;
    return v;
}
```

QUESTI SONO DETTI "FORMALI"] DEFINIZIONE

UTILIZZO ["PARAMETRI ATTUALI"

In C il passaggio di parametri avviene per valore: il valore dei parametri attuali è copiato in variabili locali della funzione

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 59 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 61 di 96

così x e y si invertono!

NON SCAMBIO GLI INDIRIZZI, MA USO GLI INDIRIZZI PER SCAMBIARE x e y

IL PASSAGGIO PER INDIRIZZO È INDISPENSABILE SE VOGLIO CHE LA FUNZIONE RITORNI PIÙ RISULTATI.

VETTORI E FUNZIONI

I VETTORI POSSONO ESSERE PARAMETRI

- PAR. FORMALI
  SI INDICA IL VETTORE CON NOME [] SENZA DIMENSIONE
- PAR. ATTUALI
  NOME DEL VETTORE SENZA []

IL NOME DEL VETTORE INDICA L'INDIRIZZO DEL 1° ELEMENTO → IL VETTORE È PASSATO PER INDIRIZZO

PERCIÒ gli elementi di un vettore passato x argomento sono SEMPRE modificati.

```
#define N 3
...
int V[N];
fun(V, N);
```

```
fun(int v[], int n)
{...
...
}
```

La funzione che riceve il vettore come argomento NON NE CONOSCE LA LUNGHEZZA, devo perciò metterla in un altro PARAMETRO

Es: SCRIVERE LA FUNZIONE nonnull() CHE RITORNI IL NUM DI NUMERI ≠ 0 DI UN VETTORE

```
int nonnull(int v[], int dim)
{
    int i, n=0;
    for(i=0; i<dim; i++)
        if(v[i]!=0)
            n++;
    return n;
}
```

Altro es:

```
#define N 5
void read(int v[], int dim);
void print(int v[], int dim);
int main() {
    int vett[N]={0};
    read(vett, N);
    print(vett, N);
    return 0;
}
void read(int v[], int dim){
    int i=0
    for(i=0; i<dim; i++){
```

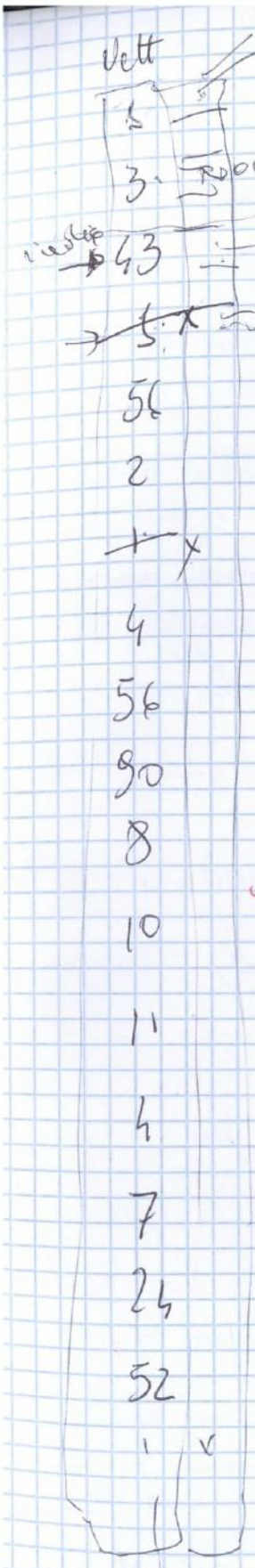
OCCHIO: ANCHE SE È void, cambia V!

QUESTO SI PUÒ FARE SOLO SE INIZIALIZZO A ZERO

BUGIA

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 61 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 63 di 96



I/O A CARATTERI

- int getchar()

legge un carattere e lo dà come risultato di getchar().

Se c'è un errore il risultato è la costante EOF (dichiarata in stdio.h)

- int putchar(<carattere>)

↓ intero

End-Of-File (in realtà è end-of-input)

Può essere generato:

- automaticamente, se si sta leggendo un file
- premendo ctrl + 'z' (MS-DOS/VMS) o ctrl + 'D' (Unix)

Es:

```
#include <stdio.h>
main(){
    int tasto;
    printf("Premi un tasto\n");
    tasto = getchar();
    if (tasto != EOF)
    {
      printf("Hai premuto %c\n", tasto);
      printf("Codice ASCII = %d\n", tasto);
    }
```

scanf e printf sono "costruite" a partire da getchar/putchar

SERVONO QUANDO NON SONO NOTE A PRIORI LE CARATTERISTICHE DELL'INPUT.

Classificazione caratteri (<ctype.h>)

		VERO	FALSO
int isalpha(char c)	se c è lettera	≠0	=0
int isdigit(char c)	se c è una cifra	≠0	=0

61

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 63 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 65 di 96

Es:
```
const int MAX = 20;
char nome[MAX+1];
printf("Come ti chiami?\n");
scanf("%s", nome);
```

printf(ANALOGAMENTE PROCEDO CON LA printf

printf
```
char nome[] = "ciccio";
printf("%s", nome);
```

Questo è un I/O FORMATTATO non perché ci siano solo caratteri alfabetici. La scanf legge caratteri fino allo spazio

Per leggere anche lo spazio uso la gets, che non si aspetta nessun formato

I/O a RIGHE

→ serie di caratteri terminata da '\n'

```
char *gets(<stringa>)
```

- la riga è fornita come <stringa>, senza il carattere '\n'
- in caso di errore, ritorna la costante NULL.

```
int puts(<stringa>)
```

- Stampa <stringa> e le aggiunge '\n'

Esempio:
```
#include <stdio.h>
#define MAX 30
int main()
{
    char stringa[MAX+1];
    char stringa2[MAX+1];
    printf("Inserisci 2 stringhe\n");
    gets(stringa);
    gets(stringa2);
    puts(stringa);
    puts(stringa2);
    return 0;
}
```

STAMPO LE 2 RIGHE ANDANDO A CAPO

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 65 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 67 di 96

#include <string.h>

char* strcat(char* s1, char* s2); CONCATENAZIONE

Es:
```
char s1[6] = "mamma";
char s2[30] = "ciao";
strcat(s2,s1); strcpy(s2,"ciao");
printf("%s\n", strcat(s2,s1));
```

ORA LA 1ª STRINGA È STATA MODIFICATA.

LA STRINGA CONCATENATA ENTRA NEL 1° ARGOMENTO, CHE È QUI s2 ⇒ s2 deve avere dimensioni sufficientemente grandi.

char*	strchr(char* s, int c);	RICERCA DI c IN s → NON s1 NON È CAMBIATA! trova il resto della stringa dal carattere trovato (compreso)
int	strcmp(char* s1, char* s2);	CONFRONTO → 0 se sono uguali, ≠0 se diverse
char*	strcpy(char* s1, char* s2);	COPIA s_2 IN s_1 ($s_1 >= s_2$)
int	strlen(char* s);	LUNGHEZZA DI s (senza "\0")
char*	strncat(char* s1, char* s2, int n)	concat un n max di caratteri
char*	strncpy(char* s1, char* s2, int n)	copia " " " " "
int	strncmp(char* dest, char* src, int n)	confronta " " " " "

Esercizio: INVERTIRE UN TESTO

```
int i,j,len;
char s[80], dest[80];
while (gets(s) != NULL) {
    len = strlen(s); j=0;
    for(i=len-1; i>=0; i--){
        dest[j] = s[i]
        j++;
    }
    dest[j] = '\0';
    puts(dest);
}
```

65

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 67 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 69 di 96

```
if ( scelta[0] = 'v')
   :
```

La gets() ci può dare un WARNING, perché è sconsigliata.

SE L'UTENTE SCRIVE 1.000.000.000 CARATTERI, TUTTI QUELLI DI TROPPO SPORCANO LA MEMORIA

⟹ SI USERÀ fgets

MEMORIZZA L'INVIO COME TALE ('\n') vd. pg 10 quaderno 2

NEL SOLITO PROGR.

POSSO USARE LA scanf() così:

```
...
printf("Scegli: v/r/q\n");
scanf("%c %*c", &car_scelta);
...
```

PER USARE STRINGA E scanf():

```
char scelta[2];
  :
printf("Scegli
scanf("%s %*c", scelta);   /* scanf("%s") NON LEGGE \n ! */
```

⟹ ATTENZIONE A MESCOLARE scanf() e getchar()!

meglio usare la gets o fgets

16-5-16

NOTA:

strcpy e strcat possono essere usate senza risultato

DI SOLITO UNA RIGA HA dimensione 80

Esercizio: UN PROGR. legga 2 stringhe e stampi i caratteri della prima che non sono nella seconda:

s1: "Olimpics"
s2: "Oio"
s3: "Impe"

```
#include <stdio.h>
#define MAXCAR 128
char *elimina(
main() {
    char s1[MAXCAR], s2[MAXCAR];
    printf("Dammi s1^a \n");
    scanf("%s", s1);
```

67

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 69 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 71 di 96

Un Esempio:

```
int x[3][5];
...
for(i=0; i<3; i++){
   for(j=0; j<5, j++){
      ...      // operazioni varie, per es. printf("%d", x[i][j])
   }
}
```

SOMMA PER RIGHE

```
for(i=0; i<N; i++)
{
float somma = 0.0;
   for(j=0; j<M; j++)
      somma = somma + mat[i][j]
   sr[i] = somma;
}
for(i=0; i<N; i++)
   printf("Somma riga %d = %f\n", i+1, sr[i]);
```

Esempio: LEGGI UNA MATRICE 5X5 E STAMPANE LA TRASPOSTA

Analisi: • HO UNA MATRICE 5X5 ⟹ NUMERO NOTO A PRIORI ⟹ SI POSSONO USARE MATRICI E VETTORI

• NON DEVO COSTRUIRE LA TRASPOSTA, SOLO STAMPARLA!

```
#include <stdio.h>
main(){
  int mat[5][5], i, j;
  printf("Inserire gli elementi per righe:\n");
  for(i=0; i<5; i++)
     for(j=0; j<5; j++)
        scanf("%d", &matrice[i][j]);
printf("\n\n");
for for(j=0; j<5; j++){
     for(i=0; i<5; i++)
        printf("%5d", matrice[i][j]);
  printf("\n");
}
}
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 71 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 73 di 96

CON SUCCESSO

Es: orario1 hh:mm → &?

```
sscanf(orario1, "%2d:%2d", ora1, minuti1);
```

Analogamente

```
sprintf(intervallo, "%d_minuti", minuti);
```

→ si vuole scrivere una stringa del tipo "60_minuti" (chiamata intervallo)

60 → variabile chiamata minuti

Es:

```
sprintf(matricola, "s%6d", n_matricola);
```

ARGOMENTI SULLA LINEA DI COMANDO

```
int main(void){

}
```

main() è una FUNZIONE, che viene chiamata dal SISTEMA OPERATIVO. Il Sist. op. può anche passare a main() degli argomenti.

NELL'INTERFACCIA

IN C SI POSSONO PASSARE INFO A UN PROGR. SPECIFICANDO DEGLI ARGOMENTI SULLA LINEA DI COMANDO

C:\> nomeprogramma <arg1> <arg2> ... <argN>

myprog

NEL MAIN()

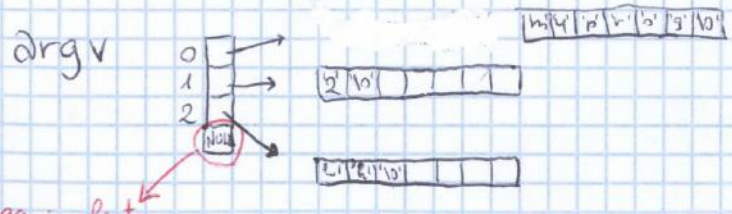
```
main(int argc, char* argv[])
```

se ho >myprog 2 -h
argc = 3 (conta anche il nome del progr.)

argc = arg counter : NUMERO DI ARGOMENTI SPECIFICATI
ne ∃ sempre almeno 1 (il nome del progr.)

argv = arg vector, vettore di stringhe

argv[0] = 1° argomento
argv[1] = 2° argomento
argv[argc-1] = ultimo argomento



NULL: equivalente del terminatore di stringa

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 73 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 75 di 96

Gli argomenti sulla linea di comando di solito indicano modalità di funzionamento alternative di 1 programma.

OPZIONI della flag o switch

di solito si indicano con -<carattere>

Es. 1

Il progr legga sulla linea di comando 2 interi N e D e stampi tutti i numeri ≤ N e divisibili per D

```
#include <stdio.h>
int main(int argc, char* argv[]){
    int N, D, i;
    if (argc!=3){
        printf("Numero argomenti non valido!\n");
        return 1;   /* qui il progr. finisce */
    }
    if (argv[1] != NULL)  N = atoi(argv[1]);
    if (argv[2] != NULL)  D = atoi(argv[2]);
    for (i=1; i<=N; i++)
        if ((i%D)==0)
            printf("%d\n", i);
}
```

ALTRIMENTI LE OPERAZIONI OPERERANNO SU STRINGHE = NULL

Es 2 Il progr m2m legga un testo e converta da maiuscole a minuscole o viceversa, a seconda dei flag:

-l -L ⇒ M→m l = lower

-u -U ⇒ m→M U = upper

-h ⇒ VIENE STAMPATO UN HELP

>> C:> m2m -l

```
#include <stdio.h>
main(int argc, char* argv[]) {
    int lowercase=0, uppercase=0;
    switch (argv[1][1]){
        case 'l':
        case 'L':
            lowercase=1;
            break;
        case 'u':
        case 'U':
            uppercase=1;
            break;
        case 'h':
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 75 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 77 di 96

Es:
```
struct studente {
    char cognome[30];
    char nome[30];
    unsigned int matricola;
    float media;
};
```
→ ATTENTO, eh!

⇒ struct studente sarà il nome del TIPO, ma non della variabile!!

```
struct studente mario; /* vuol dire: mario è una variabile di tipo struct studente */
```

La struct <nome> di solito si pone PRIMA DEL MAIN

IN ORDINE:
```
#include ...
#define ...
struct <nome>{...
}
<prototipi>
main(){
...
}
```

Es: I NUMERI COMPLESSI
```
struct complex{
    double re;
    double im;
}
...
struct complex num1, num2;
```

Per accedere ai singoli campi: <variabile>.<campo>

Nell'es. precedente:
```
num1.re = 0,33; num1.im = -0,43943;
```
Oppure
```
scanf("%d", &num.re);
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 77 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 79 di 96

Soluz. del lab. 9, es. 2

Nel passare una matrice,

SI DA' UNA DELLE 2 DIMENSIONI

27-5-14

PER ANDARE SULLA LINEA DI COMANDO:

Start → Esegui → cmd.exe

Programmi → Accessori → Prompt dei comandi

- dir per visualizzare stato di file nella cartella corrente
- cd (change directory) per muoversi nella cartella

Il file eseguibile è .exe → lo trovi nella cartella bin → debug

<DIR> vuol dire CARTELLA

- ~~esta~~ cd .. per tornare indietro di livello.

ALTRO MODO PER ESEGUIRE DA LINEA DI COMANDO:

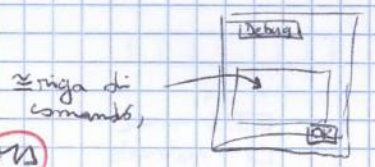
SU CODEBLOCKS

Project

↓

Set programs' arguments

↓



MA

IL NOME DEL PROGR. È IMPLICITO

Poi vado su RUN

STRUCT:

- NON SI PUÒ fare un confronto diretto tra 2 variabili dello stesso tipo struct, per es. studente1 == studente2

⇒ Il CONFRONTO si fa CAMPO A CAMPO

Es: complex z1, z2;

if((z1.re == z2.re) && (z1.im == z2.im))

... OK

- COME PER I VETTORI, POSSO FARE LA INIZIALIZZAZ. CUMULATIVA, con { }. I VALORI MANCANTI VARRANNO 0 PER VALORI SCALARI, NULL PER I VETTORI

5

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 79 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 81 di 96

30 - 5 - 14

I FILE

- FILE BINARIO
  - FORMATO "PROPRIETARIO" (sistemi riconosciuto solo dalla casa che l'ha creato): .doc, .xls
  - FORMATO "STANDARD": .pdf, .jpeg, .zip
- FILE ASCII
  - TESTO PURO: .txt (estensione)
  - TESTO FORMATTATO (grassetto, corsivo...): .html
  - LINGUAGGIO FORMALE (programmazione): .c

FILE SEQUENZIALI

ACCESSO BUFFERIZZATO

BUFFER = memoria interna al sistema

VANTAGGI:

- MAGGIORE ASTRAZIONE
- POSSIBILITÀ I/O FORMATTATO

Con l'I/O non bufferizzato si ha l'ACCESSO DIRETTO A LIVELLO BINARIO UN CARATTERE PER VOLTA.

C vede il file come un FLUSSO (stream) SEQUENZIALE di byte.

⇒ NESSUNA STRUTTURA PARTICOLARE → LA STRUTTURA È A CARICO DEL PROGRAMMATORE.

NOTA:
- non posso leggere all'indietro
- non posso saltare a un punto specifico del file

BANANA

Accesso TRAMITE UNA VARIABILE DI TIPO FILE* definita in stdio.h

DICHIARAZIONE:

FILE* <file>

contiene un insieme di variabili che permettono l'accesso ai tipi.

Quando il progr. viene eseguito, in automatico sono attivati 3 file:

- stdin (input) → associato alla tastiera
- stdout → associati al video
- stderr (error) → associati al video

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 81 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 83 di 96

CHIUSURA

si rompe la connessione tra file logico e fisico

```
int fclose(FILE* <file>);
```

RITORNA: 0 OK; EOF errore

Esempio:

```
FILE* fp;
...
/* apro file 'testo.dat' in lettura */
fp = fopen("testo.dat", "r");
if (fp==NULL)
    printf("Errore nell'apertura\n");
else
{
    /* qui posso accedere a 'testo.dat' usando fp */
}
...
fclose(fp);
```

CONTROLLO DA FARE SEMPRE.

Nel controllo è bene fare così:

```
if (fp == NULL) {
    printf("Errore apertura\n");
    return 1;
}
```

ATTENTO! Qui non devi chiudere il file, visto che il file non è stato aperto

Si può fare un controllo anche su fclose

```
int r = fclose(fp);
if (r == EOF){
    printf("Errore chiusura\n");
    return 1;
}
```

LETTURA DI UN FILE:

leggi riga/carattere

POSIZ. INIZIALE (1° CARATTERE) → POSIZ. INTERMEDIA (n-ESIMO CARATTERE) → POSIZ. FINALE (ULTIMO CARATTERE) ⇒ CHIUSURA

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 83 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 85 di 96

LETTURA FORMATTATA

int fscanf(FILE* <file>, char* <formato>, ...);

qui è il 1° parametro.

Come scanf() + parametro addizionale x il file

RITORNA → NUM. CAMPI CONVERTITI
→ EOF

scanf(...) ⟺ fscanf(stdin, ...)

SCRITTURA FORMATTATA

int fprintf(FILE* <file>, char* <formato>, ...);

Come printf + parametro x il file

RITORNA: → NUM BYTE SCRITTI
→ EOF

printf(...) ⟺ fprintf(stdout, ...)

ALTRE FUNZIONI

FILE freopen(

fopen su file già esistente

int feof(FILE* < >);

→ 0 PUNTATORE NON SULLA FINE DEL FILE
→ 0! IL CONTRARIO

Ad ogni file è associato un BUFFER e un puntatore nel buffer.

La posiz. del puntatore può essere manipolata.

void rewind(FILE* <file>)

posiziona il puntatore a inizio file

SCHEMA GENERALE DI LETTURA FILE

leggi un dato dal file;
finche' (non è finito il file)
{
elabora il dato;
leggi un dato dal file
}

QST CONDIZ. SI PUO' REALIZZARE USANDO I VALORI RESTITUITI DALLE FUNZIONI DI INPUT (EOF o NULL)

meglio →

oppure USANDO feof()

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 85 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 87 di 96

Riepilogo di I/O

`scanf("%c", &var);` legge tutto (anche ' ' e '\n')

`int getchar()` → carattere letto / EOF

`scanf("%d", &var);` non legge ' ' né '\n'

`int putchar(ch);`

`scanf("%s", str)` non legge ' ' né '\n'

`char* gets(str)` → NULL / stringa — legge '\n' e lo rimpiazza con '\0'

`int puts(str);` stampa e aggiunge '\n' ⇔ `printf("%s\n", str);`

I/O a caratteri

I/O a righe/stringhe

`int getc(FILE* <fp>)`

`int fgetc(FILE* <fp>)` → carattere / EOF

`int putc(int ch, fp)`

`int fputc(int c, FILE* <fp>)` → carattere / EOF

`char* fgets(str, n, FILE* <fp>)` → stringa / NULL — legge '\n' come tale (max n-1 caratteri)

`int fputs(str, FILE* <fp>)` → ultimo carattere / EOF — non aggiunge '\n'

`int FscanF(FILE* <fp>, char* <formato>, ...);` → num. campi convertiti / EOF

`int fprintF(FILE* <fp>, char* <formato>, ...);` → num. byte scritti / EOF

caratteri

righe/stringhe

FILE

3-6-14

Es. file non formattato:

```
c = getc(fp);
while (c != EOF){
  /*elabora c*/
  c = getc(fp);
}
```

OPPURE

```
while ((c = getc(fp)) != EOF){
  /*elabora c*/
}
```

Altro es:

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 87 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 89 di 96

printf() nel profondo

short int	%hd %d
long int	%ld
unsigned int	%u
unsigned long int	%lu

printf("%1d", 13); stampa cifra 13

↳ DIM. MINIMA

scanf() approfondita

scanf salta tutti i caratteri di spaziatura.

Si ferma al primo carattere non di spaziatura o End-Of-File

Legge gli spazi solo se c'è %c

scanf cerca di convertire i caratteri secondo il formato specificato.

scanf("%2s", v)	input	
	1234xyz	v = "12"

scanf RITORNA UN NUMERO INTERO

num. elementi (%) letti

- non conta %*
- non conta quelli non letti xK l'input non ha i caratteri desiderati
- non conta " " " " " " " " " formato troppo presto
  - End-Of-File per fscanf
  - Fine stringa per sscanf

EOF se l'input era già in condiz. di EOF all'inizio della lettura.

%[r]	legge solo 'r'
%[a-zA-Z]	" " l'alfabeto
%[^x]	" qualunque sequenza che non contenga x
%[^\n]	" fino a fine riga

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 89 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 91 di 96

Un esempio:

```
main(){
   int a, *aptr;
   a=7;
   aptr = &a;
   printf("L'indirizzo di a e' %p\n", &a);          indir a
   printf("Il valore di aptr e' %p\n", aptr);       ind_a
   printf("Il valore di a e' %p\n", a);             a
   printf("Il valore di *aptr e' %p\n", *aptr);     a
   printf("&*aptr = %p ---- *&aptr = %p\n", &*aptr, *&aptr);   ind_a
```

CODICE X STAMPARE LA VARIABILE IN FORMATO PUNTATORE

OPERAZIONI SUI PUNTATORI:

SOLO ALCUNE SONO VALIDE:

- INCREMENTO / DECREMENTO

  `int *px;  px++;` o `px--;`

- SOMMA / SOTTRAZ. DI UN VALORE INTERO

  `int *px;  px+=3;` o `px-=5;`

- SOTTRAZ. TRA PUNTATORI (NON LA LORO SOMMA) DELLO STESSO TIPO

  `int *px, *py;  px-py;`

  IL RISULTATO È LA DISTANZA TRA I 2 INDIRIZZI

- ASSEGNAZ. DI UN PUNTATORE A UN ALTRO DELLO STESSO TIPO

  `int *px, *py;  px=py;`

- CONFRONTO TRA PUNTATORI DELLO STESSO TIPO

  `int *px, *py;  px==py;` o `px!=py;` o `px>py;` ecc.

  TIPICO: `px==NULL`

- ASSEGNAZ. DI UN PUNTATORE AD UN VALORE COSTANTE (sconsigliato)

  `int *px;  px=5;`

  l'accesso potrebbe essere vietato!

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 91 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 93 di 96

Esempi:

```
long *p1, *p2;
int j;
char *p3;

p2 = p1 + 4;        // OK
j = p2 - p1;        // OK → j=4
j = p1 - p2;        // OK → j=-4
p1 = p2 - 2;        // OK
p3 = p1 - 1;        // NO! TIPI DIVERSI!
j = p1 - p3;        // NO! TIPI DIVERSI DI PUNTATORI!
```

VETTORI E PUNTATORI

IL NOME DEL VETTORE È L'INDIRIZZO DEL SUO 1° ELEMENTO

```
int a[8], *aptr;
aptr = a;
```

così aptr punta il primo elemento di a.

$$a[i] \iff *(aptr+i)$$

2 MODI PER ACCEDERE A 1 ELEMENTO DEL VETTORE:

1- USO DELL' ARRAY (tramite un INDICE)

2- USO DI UN PUNTATORE (tramite ~~SPOSTAMENTO~~ OFFSET)

Pertanto:

```
int a[], *aptr = a;

a[i]
*(a+i)
aptr[i]
*(aptr+i)
```

4 modi per accedere all'elemento.

Esempio:

```
main(){
  int b[] = {1,2,3,4};
  int *bptr = b;
  int i, offset;
```


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 93 di 96

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 95 di 96

Si può così sfruttare appieno la libreria delle stringhe

ES: char* strchr(char *s, int c); // trova la 1ª occorrenza di c in s

```
char *p, *string = "stringa di prova";
int n, i;
if ((p = strchr(string, ' ')) != NULL){
    printf("La stringa prima dello spazio e': ");
    n = p - string;
    for (i=0; i<n; i++) putchar(string[i]);
    putchar('\n');
    printf("La stringa dopo lo spazio e': %s\n", p+1);
}
else{ printf("Non ho trovato ' ' nelle stringa %s\n", string);
```

/* N.B!! */

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 95 di 96