NUMERO: 1004 DATA: 01/07/2014

A P P U N T I

STUDENTE: Renis

MATERIA: Informatica

Prof. Mezzalama

INFORMATICA → scienza che rappresenta e manipola le informazioni.

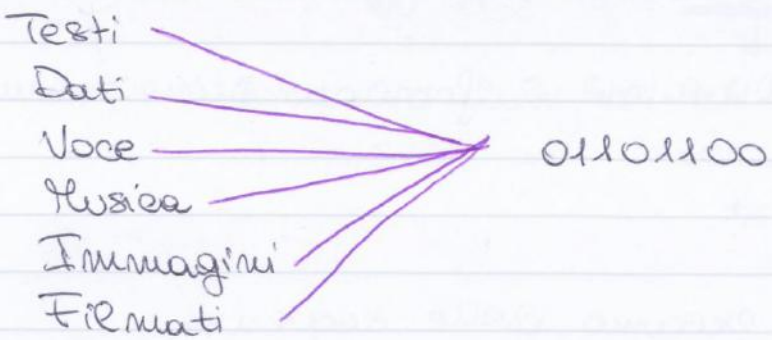
IL COMPUTER MODERNO

1942 - 1957	1ª generazione	= tubi a vuoto
1958 - 1963	2ª generazione	= transistori
1964 - 1980	3ª generazione	= circuiti integrati
1980 - oggi	4ª generazione	= circuiti VLSI

Informazione = è la conoscenza (concetto astratto)
Dati = modalità con cui questo tipo di informazione si rappresenta.

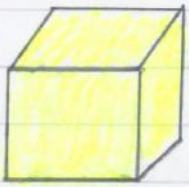
I DATI DIGITALI: TUTTO DIVENTA "BIT"

Tutta l'informazione che prima era codificata attraverso diversi alfabeti, ora viene codificata attraverso un solo alfabeto (= sequenze di bit) grazie all'evoluzione del mondo digitale. ↓
sequenze di zeri e uno



Per scrivere un programma, devo fornire dei dati al mio calcolatore, che è in grado di capire solo sequenze binarie.

BIT (0, 1)
(Binary digit) ⇒ (= cifra binaria)



Il bit è troppo piccolo per rappresentare anche l'informazione più semplice, quindi sostanzialmente l'unità con cui si raccolgono i bit è un ottetto (8 bit), chiamato BYTE.

NOTAZIONE: bit → b
byte → B

I Byte possono essere aggregati in più corpi.
Per rappresentare i numeri interi servono due byte, oppure possono essere aggregati in 4 byte.

sono la lunghezza su cui si rappresentano i $n \in \mathbb{R}$ all'interno degli elaboratori.

CHI PENSA MEGLIO

1 PC = 500 miliardi di calcoli
1 server = 2'000 miliardi di calcolo

Umano = 1000 miliardi di neuroni
Cane = 100 milioni di neuroni

Non bisogna confondere l'intelligenza con la capacità di calcolo.

memorizzazione (come riesco a dimensionare la memoria?)

- si usano le potenze di due invece che di dieci (perché il calcolatore ragiona in binario)
  - chilo K 2^{10} migliaio
  - mega M 2^{20} milione
  - giga G 2^{30} miliardo
  - tera T 2^{40} mille miliardi
  - peta P 2^{50} milione di miliardi
  - exa E 2^{60} miliardo di miliardi

LE MISURE DELL'INFORMAZIONE DIGITALE

Pagina dattiloscritta (di un libro) in word → memoria 2 KB

Film → 1 GB

1000 GB = 1 T (terabyte)

↓

formulò l'ipotesi di una macchina che potesse essere programmata e faceva dei calcoli.
La espose a Torino all'Accademia delle Scienze.

PROBLEMA = non aveva a disposizione gli strumenti adatti.
Non c'era ancora la tecnologia elettronica.
Egli costruì il suo calcolatore con la tecnologia meccanica.

Primo calcolatore elettronico (1943-1945) → nato per intercettare le bombe tedesche sul territorio inglese.

TRANSISTOR → ha permesso di rendere minuscoli i dispositivi elettronici.

Intel → ditta americana → produce (cip) (microprocessori)
li producono in Asia, ma li progettano in California

Toshiba → grande produttore di (RAM) (memoria)

la T. esterna, più facile è la dissipazione del calore.

COSA IMPARIAMO IN QUESTO CORSO?

Impariamo a programmare. Cerchiamo di realizzare dei programmi per risolvere dei problemi.

DIFFICOLTÀ

I punti critici nello sviluppo di un progetto risiedono nello:
- sviluppo di una soluzione "informale"
- formalizzazione di una soluzione.

La soluzione di un problema passa generalmente attraverso lo sviluppo di un algoritmo.

Algoritmo → descrizione precisa di una sequenza finita di azioni che devono essere eseguite per giungere alla soluzione di un problema.

Algoritmi e vita quotidiana

1. metti l'acqua
2. accendi il fuoco
3. aspetta
4. se l'acqua non bolle torna a 3
5. butta la pasta
6. aspetta un po'
7. assaggia
8. se è cruda torna a 6
9. scola la pasta

STADI DI SVILUPPO DI UN PROGRAMMA

1. Scrittura di un programma utilizzando un linguaggio di programmazione.
2. Traduzione di un programma in un formato comprensibile al calcolatore.

- PROBLEMA
- IDEA (soluzione)
- ALGORITMO (soluzione formale)
- PROGRAMMA (traduzione dell'algoritmo in una forma comprensibile ad un elaboratore elettronico).
- TEST
- DOCUMENTAZIONE

FORMALIZZAZIONE DELLA SOLUZIONE

Soluzione

- informale: descrizione a parole
- formale: descrizione in termini di sequenza di operazioni elementari

la differenza sta nel modo di rappresentare un algoritmo.

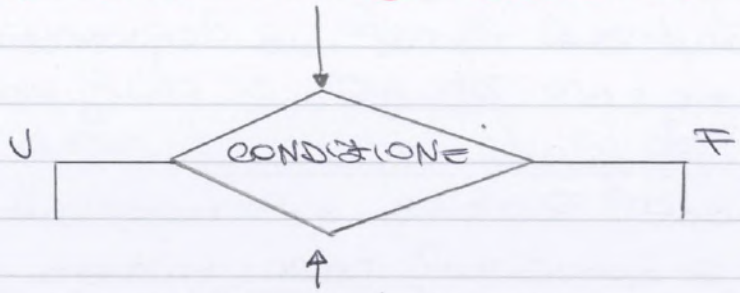
SCRITTURA DEL PROGRAMMA

- Un sorgente C è un normale file di testo
- Si utilizza un editor di testi
  - Blocco note
  - EDITOR SPECIALIZZATI PER PROGRAMMATORI

    ↓

    - colorazione ed evidenziazione della sintassi
    - identazione automatica
    - attivazione automatica della compilazione
    - identificazione delle parentesi corrispondenti
    - molti disponibili, gratuiti e commerciali,

BLOCCO DI DECISIONE (o BLOCCO DI IF)

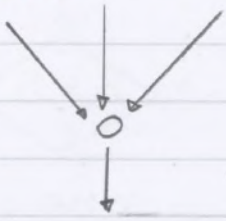


condizione booleana

(se succede questo vai di qui, se succede quest'altro vai di là)

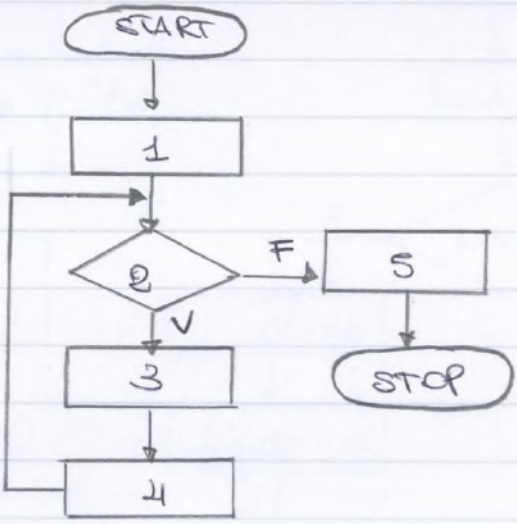
è una condizione che ha come risposte V o F.

BLOCCO DI CONNESSIONE



È il punto in cui convergono più strade provenienti da più blocchi.

Esempio

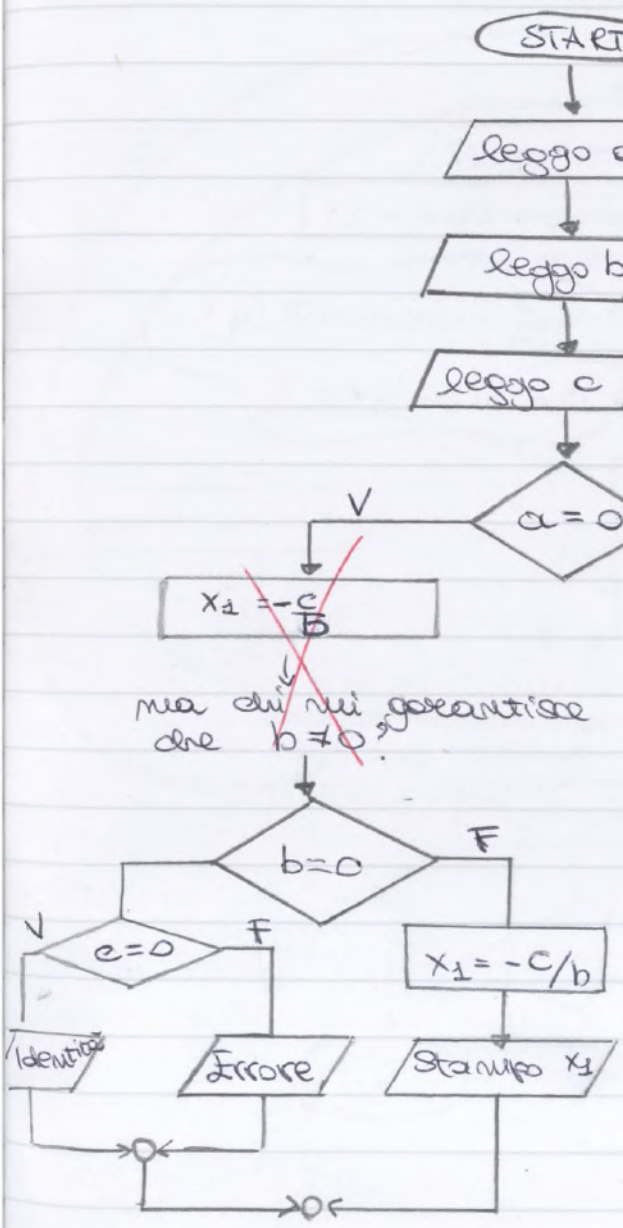


Questo potrebbe essere un algoritmo che non finisce mai se nella condiz. 2 non trovo una risposta falsa

Ho una rappresentazione grafica di come si svolge l'algoritmo.

COMMENTO

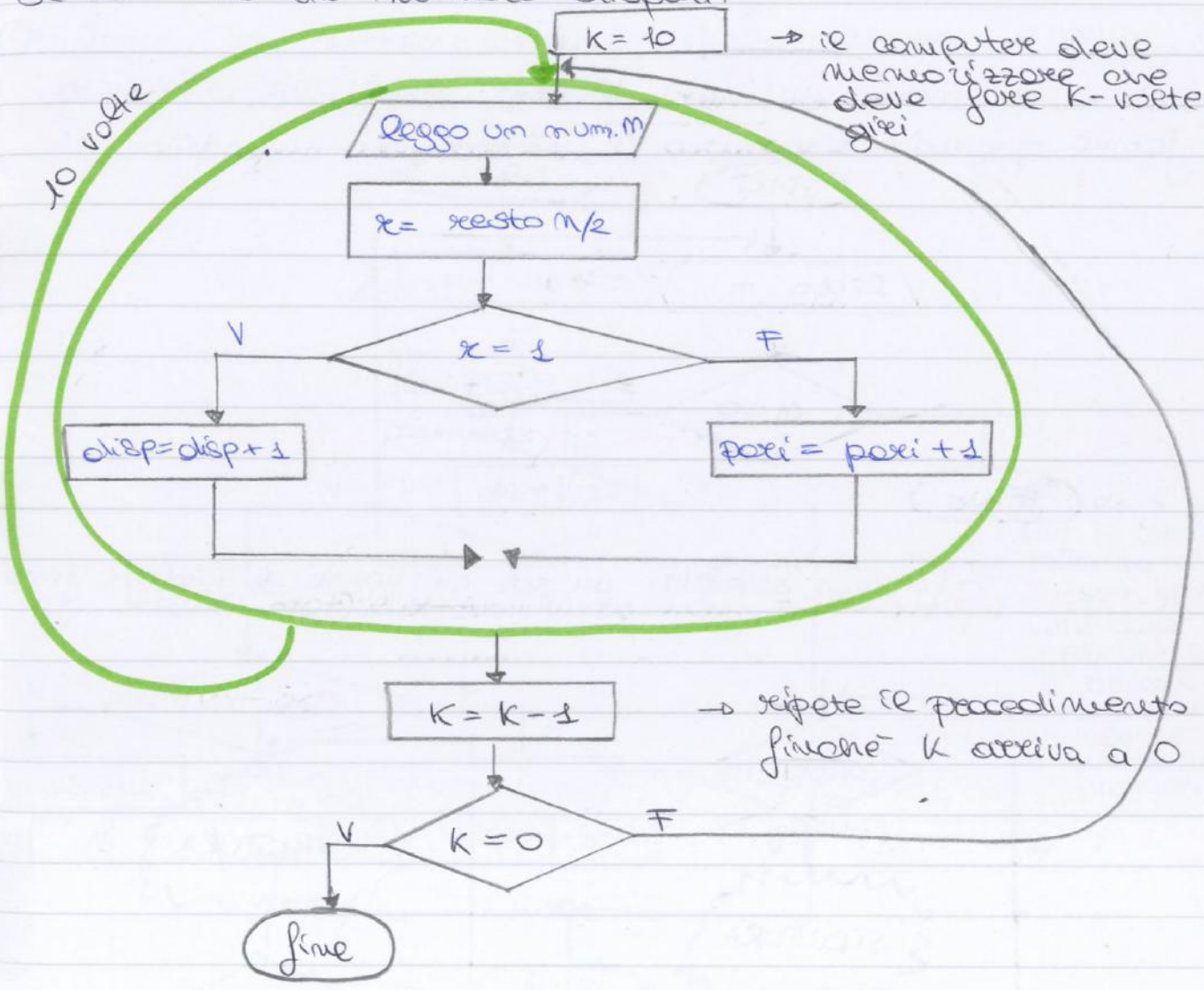
Questo algoritmo è sbagliato. Perché non ho detto che valori hanno a, b, c. Quindi se ad es. a vale 0, ho un numero fratto 0, che dà ∞.
Ci sono delle condizioni limite da rispettare.
Modifico l'algoritmo.



b=0 → dal punto di vista del calcolo non mi comporta niente.
c=0

COMMENTI

Se invece di avere 10 numeri ne avessi 10'000 come farei? Diagramma di flusso "infinito"? NO.
Nota che tutti i blocchi ripetono la stessa cosa: leggo un numero n, calcolo il resto di n/2, mi chiedo se r è = 1, se è F vuol dire che è un numero pari, se è V è un numero dispari.



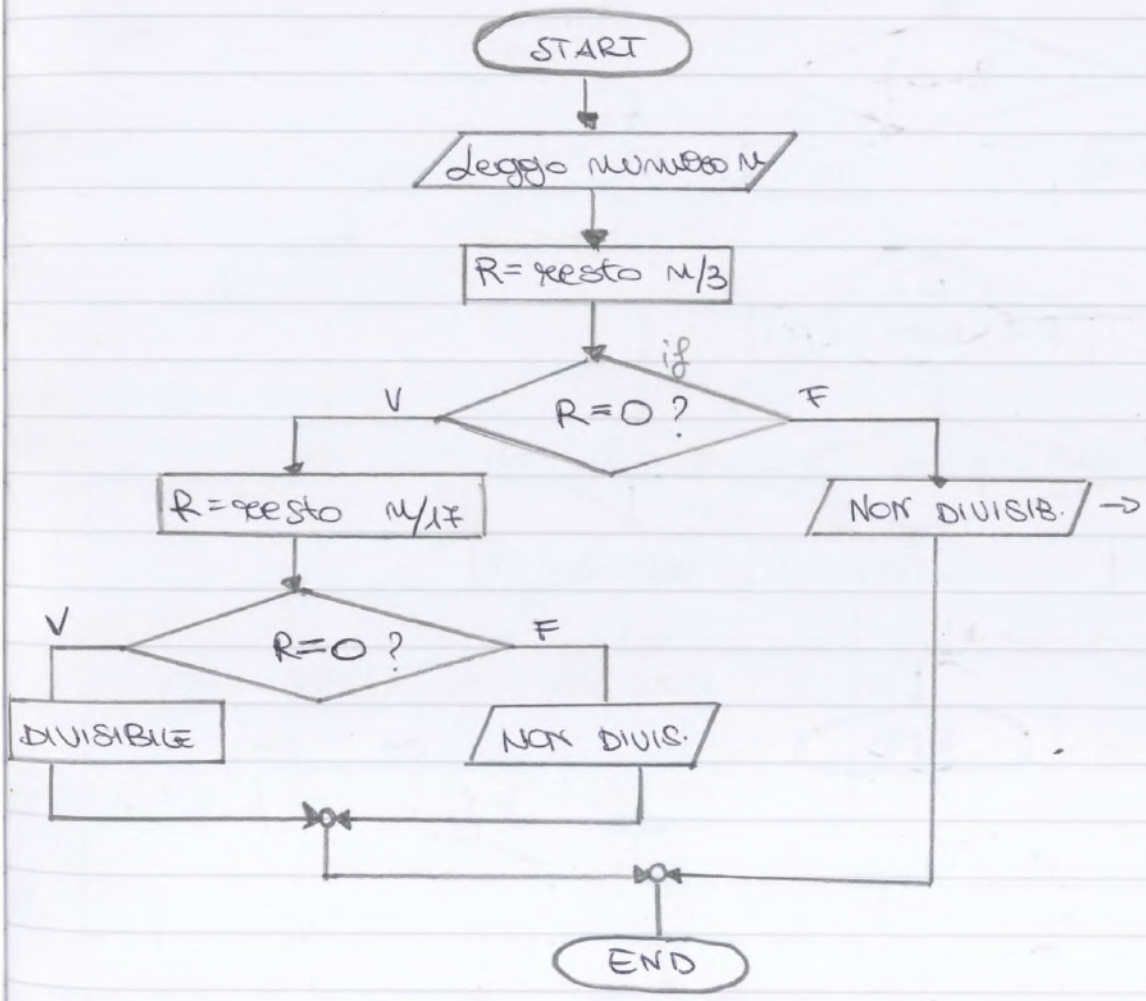
ITALIANO

TEOREMA DI BÖHM - JACOPINI

"Qualunque diagramma di flusso è sempre trasformabile in un diagramma di flusso strutturato equivalente a quello dato".

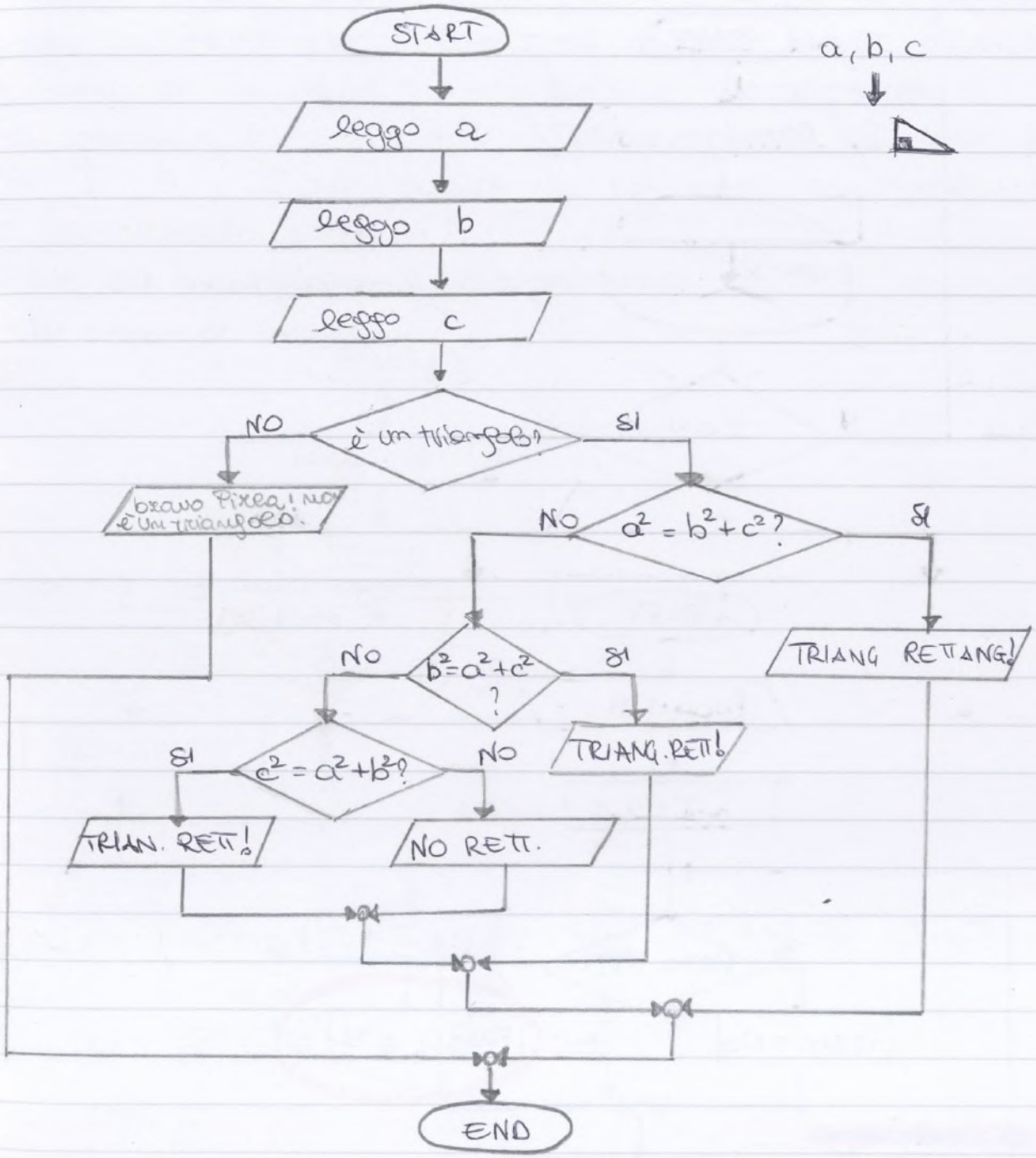
ESERCIZI FLOW-CHART

1. Leggo un numero naturale dalla tastiera e verifico se è divisibile per 3 e per 17.



se non è divis per 3 abbiamo risparmiato il prob e non dobbiamo verificare per 17 perché il prob di calcolo quando è divisib contemp. per entrambi i numeri.

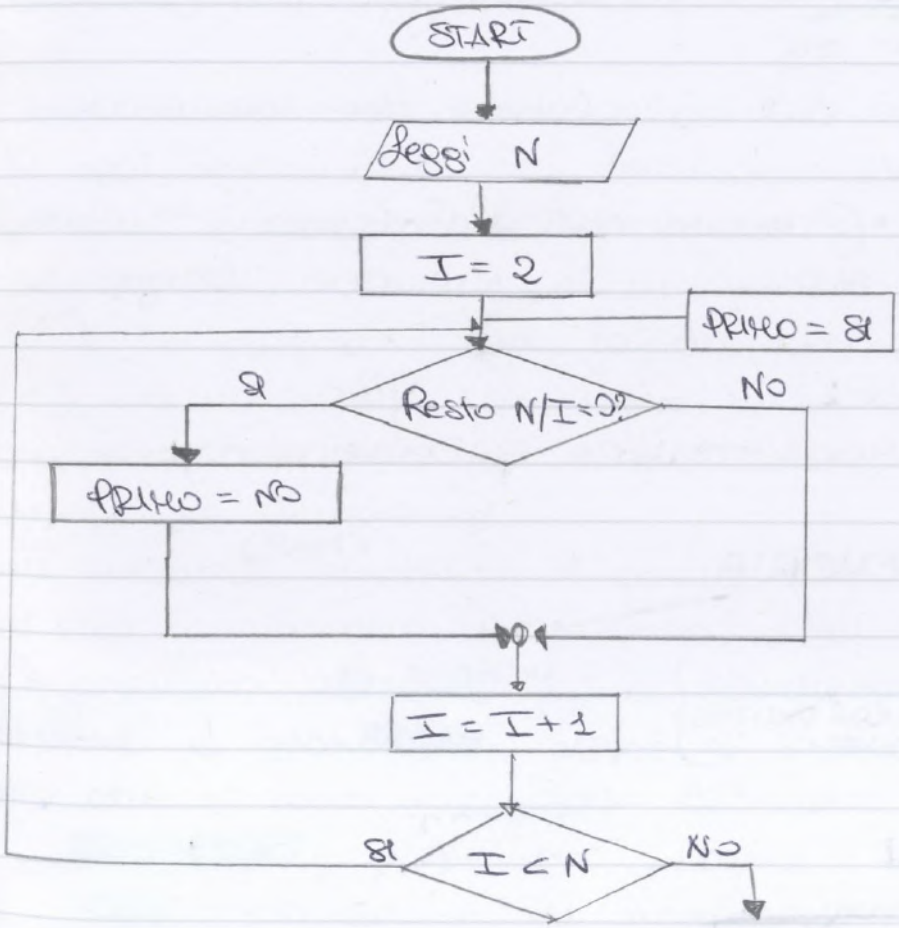
② leggo da tastiera 3 numeri, corrispondenti a 3 lati di un possibile triangolo. Il triangolo è rettangolo.



COMMENTI

Supponiamo di avere il n°=6 che non è un numero primo. Prendo il 2, faccio 6/2, R=0 e il diagramma mi porta a PRIMO = NO. I = I+1 quindi da 2 diventa 3. Ritorno su e 6/3 = 2, R=0, ed è giusto. Poi prendo 4. 6/4 = 1, R=2. R≠0 avviene PRIMO = SI Poi I = 5, I > N quindi mi porta a concludere che 4 è un numero primo. (F)

Come si modifica il diagramma?



La memoria contiene → DATI → ricevuti dall'ingresso ed elaborati.

→ PROGRAMMI → che l'unità centrale di elaborazione deve eseguire.

La memoria centrale è veloce ma di dimensioni ridotte. Per cui abbiamo bisogno di una memoria di massa che contenga una molteplicità elevata di informazioni. Questa memoria è data dai dischi.

I CHIP FONDAMENTALI

Se noi apriamo il PC troviamo dei dispositivi elettronici (chip = "briciola") che svolgono le funzioni di cui si è parlato prima.

C'è un dispositivo che fa da CPU (unità di elaborazione) normalmente chiamato microprocessore, perché è un processore, ossia un elaboratore, e micro perché è di piccole dimensioni.

La memoria centrale (RAM) è anche essa fatta da un insieme di chip e ciascuno memorizza delle informazioni.

Se andiamo ancora più a fondo prendendo un microprocessore e guardiamo come è fatto al suo interno, vediamo che ci sono una serie di unità funzionali di cui:

REGISTRO → memoria locale, piccola ma molto veloce

ALU (arithmetic logic unit) → porzione di hardware che serve a fare i calcoli.

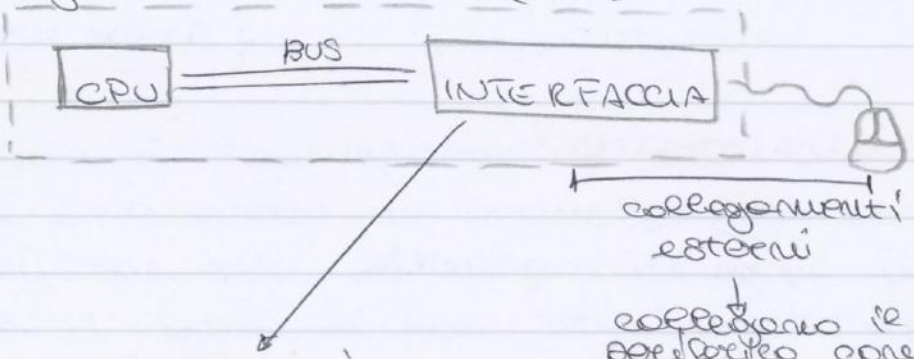
Se potenziata riesce a fare i calcoli anche su numeri con la virgola.

FPU (floating point unit) → calcolatore per i numeri reali.

- Più è potente un elaboratore più il numero di bit trasferibili aumenta (per quanto riguarda il DBUS).
- Quanto più grande è l' ABUS tanto maggiore è la memoria.

DISPOSITIVI PERIFERICI

Sono collegati al sistema (CPU) mediante un interfaccia



prende i dati grezzi e li traduce in bit e tramite i bus li porta al CPU.

collegano il dispositivo periferico con il calcolatore possono essere di tipo wireless o con il filo.

- non ci sono bit in senso stretto.
- contengono i dati che devono essere inviati o ricevuti.

ci sono bus di dati e bus di controllo.

controllano lo stato del sistema. Determinano la possibilità e la correttezza del trasferimento

I COLLEGAMENTI ESTERNI

Sono di diverso tipo:

- USB → sono uno standard meccanico.
- orientati alle memorie di massa → SATA, SCSI, EIDE

CHIP → realizzato su un supporto di silicio (sabbia)

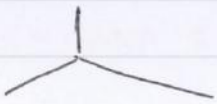
Smontiamo un computer:

- PIASTRA ELETTRONICA
- CONNETTORI ESTERNI (USB) → collegano i dispositivi periferici
- CONNETTORI INTERNI → servono per aggiungere memoria

MEMORIA

La memoria è organizzata in celle. Ad ogni cella di memoria è associato un indirizzo (numerico) per identificarla univocamente. Le celle contengono un numero variabile di byte in base all' "architettura" di quel calcolatore.

Memoria interna:
- all' interno dell' elaboratore
- è allo stato solido (chip)
- solitamente è volatile → quando la spegniamo perde l'informazione
- veloce (nanosecondi, 10^{-9} s)
- quantità limitate (qualche GB).
- non rimovibile
- costosa (0,1€ / MB)



RAM ROM

l' informaz. resta permanente anche quando spengo il dispositivo.

Memoria esterna:
- all' esterno dell' elaboratore
- talvolta rimovibile
- non elettronica (es. magnetica, ottica)
- permanente

Memorie ROM

Si dicono "memorie a sola lettura".

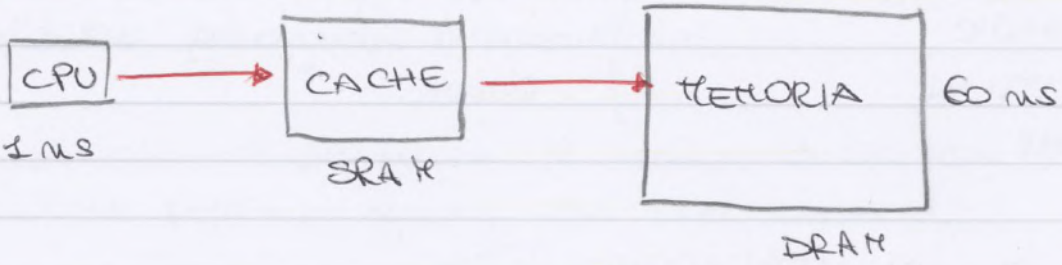
i dati sono scritti in fabbrica

i dati possono essere scritti con un tempo più lungo della velocità di un processore e possono essere scritti anche da un computer.

EEPROM

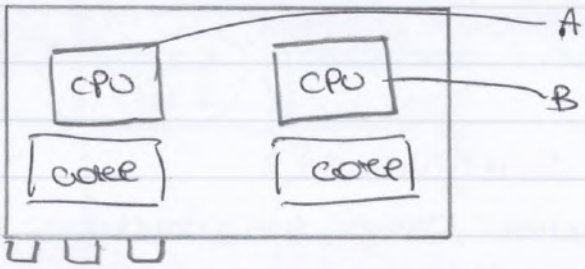
(sono flash memory le pen-drive).

Cache → è una memoria veloce intermedia



Nel 99 % dei casi la cache velocizza il processo.

Multi-core: 2 o più processori inclusi nello stesso chip ↓



⇔ ha tanti CPU

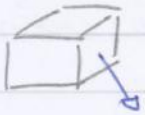
Processori Intel – i3
i5
i7

DATI

I programmi manipolano dei dati.
I dati sono la rappresentazione di informazioni.
↓
Caratteristiche:

- NOME (identificatore)
- INTERPRETAZIONE (tipo)
- MODALITÀ DI ACCESSO (costante) o (variabile)

I dati sono contenuti in delle scatole.



PIPPO
interi → 16 bit

Ogni scatola ha una dimensione.

Identificatore → definisce il nome della scatola
le scatole possono avere dei contenuti

variabili — costanti

PROGRAMMA

DICHIARAZIONE DI DATI

In C, tutti i dati devono essere dichiarati prima di essere usati.

- La dichiarazione di un dato richiede:
  - l'**allocazione** di uno spazio in memoria atto a contenere il dato.
  - l'**assegnazione** di un nome a tale spazio in memoria.
- Occorre specificare
  - NOME
  - TIPO
  - MODALITÀ DI ACCESSO

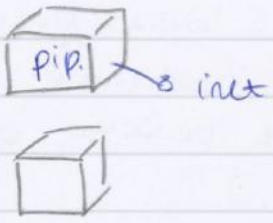
ES.

<tipo> <nome> (;)

dimensione

int pip;

int mm;



oppure

int pip, mm;

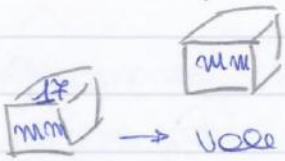
TIPI BASE

- char → 8 bit → caratteri → 'a' scatola con (1) carattere.
- int → interi
- float → reali (singola precisione)
- double → reali (doppia precisione)

ES.

int mm;

int mm = 17;



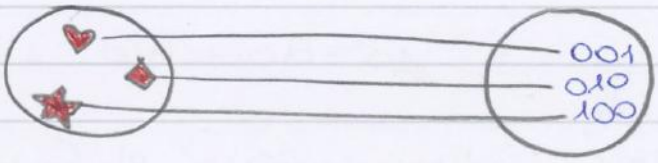
→ non ha un valore specifico

→ vale 17

Lez. 4 : CODICI NUMERICI

Codificare l'informazione → passare da una realtà ad una rappresentazione della stessa mediante un linguaggio.

Un codice binario associa ad ogni elemento una sequenza di bit.



Dati n bit si hanno 2^n combinazioni binarie differenti

ES. 5 bit → 2^5 codifiche diverse ⇒ con 5 bit posso rappresentare 32 elementi — oggetti diversi

Per codificare K oggetti occorre un numero di bit pari a:

$$n \geq \log_2 K$$

log in base 2 perché siamo in un sist. binario

con n intero

$$n = int(\log_2 K)$$

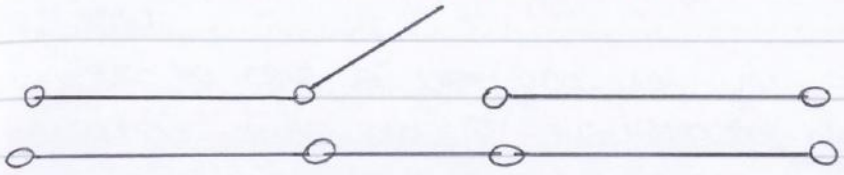
CODIFICA DEI NUMERI

SISTEMI DI NUMERAZIONE

Il sistema di numerazione del mondo occidentale è:
- decimale → ha 10 cifre (da 0 a 9) → numeri = combinazioni di queste cifre.
- posizionale

Noi ragioniamo in termini di numeri:
- relativi (int)
- reali
- naturali

BIT E INTERRUTTORI



APERTO = 0

CHIUSO = 1

↓

la loro grandezza si misura in nanometri

IL SISTEMA BINARIO

BASE = 2

CIFRE = {0, 1}

ES

$$101_{(2)} = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 4 + 0 + 1 = 5_{(10)}$$

CODIFICA DEI NUMERI

Da base 10 a base 2

ES.

$$25_{(10)} \rightarrow ?_{(2)}$$

Metodo delle successive divisioni

Least Significant Bit → ultima cifra

25 : 2

Q = 12 : 2

25 : 2 → R = 1 ← Q = 6 : 2

12 : 2 → R = 0 ← Q = 3 : 2

6 : 2 → R = 0 ← Q = 1 : 2

3 : 2 → R = 1 ← Q = 0

1 : 2 → R = 1 ← prima cifra → Most Significant Bit

$$25_{(10)} \rightarrow 11001_{(2)}$$

OVERFLOW

È l'errore che si verifica in un sistema di calcolo automatico quando il risultato di un'operazione non è rappresentabile con la medesima codifica e numero di bit degli operandi.

Es.

1	0	2	+
0	5	1	=
1	5	3	
5	7	1	+
5	1	1	
1	0	8	2

decimale

→ il risultato è sbagliato

resta fuori

→ anomalia ⇒ overflow

IL SISTEMA OTTALE

- base = 8

↓

utile per scrivere in modo compatto i numeri binari (in blocchi di 3)

Es.

$2^2\ 2^1\ 2^0$

101 110 101 001 (2)

3 bit → 2^3 combinazioni → 8 cifre diverse

vale: 5 + 6 + 5 + 1 = $17_{(10)}$

→ in sistema ottale

011 → 3 2 1 → 031

lez. 8 : NUMERI RELATIVI INTERI

- Il segno dei numeri può essere : { - positivo (+) / - negativo (−) }
- E' quindi facile capire in binario:
  - modulo e segno → codifica
  - complemento a due → codifica

CODIFICA "MODULO E SEGNO"

- Un bit per il segno
  - 0 = segno positivo (+)
  - 1 = segno negativo (−)
- N − 1 bit per il valore assoluto (anche detto modulo)

SEGNO	MODULO
1 bit	N − 1 bit

ES. Codifico su 4 bit

$+ 3_{10} \rightarrow 0011_{MBS}$

$- 3_{10} \rightarrow 1011_{MBS}$ → modulo e segno

{ 3 bit modulo / 1 bit segno }

$0000_{MBS} \rightarrow +0_{10}$

$1000_{MBS} \rightarrow -0_{10}$

$1010_{MBS} + \Rightarrow -2_{(10)}$

$0010_{MBS} = \Rightarrow +2_{(10)}$

$1100 \Rightarrow -4 \rightarrow$ 0.o. $-2+2 \neq -4$!

↓

ERRORE: otteniamo addizionato anche i bit del segno

⇓

$1(-)\ 010_{MBS} +$

$0(+)\ 010_{MBS} =$ si annullano a vicenda

$000 \Rightarrow 0$

Supponiamo di dover sommare +5 e -5

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

ES. 2.

Supponiamo di avere 1101, sequenza in complemento a 2. Quanto vale questa sequenza?

1	1	0	1	→ 2'c ?
-2^3	2^2	2^1	2^0	

1° modo

$$1\cdot(-2^3)+1\cdot(2^2)+0\cdot(2^1)+1\cdot(2^0)$$

$$-8+4+1=-3_{(10)}$$

2° modo → COMPLEM. → se ho +N trovo -N e viceversa

1101 (-3)

1. inverto: → gli 0 scambiati con gli 1

0010

2. Sommo 1:

```
  0 0 1 0 +
  0 0 0 1 =
---------
  0 0 1 1
  2⁰ 2⁰ 2¹ 2⁰
```

→ +3 (10) ⇒ quindi il numero di partenza è -3.

VIRGOLA MOBILE

RAPPRESENTAZIONE DEI NUMERI REALI

Il sistema binario è posizionale → contiene delle cifre anche a destra del segno.

ES.

$$32,27 = 10^{1} \; 10^{0} \; 10^{-1} \; 10^{-2}$$

$$0101.11 = 2^{3} \; 2^{2} \; 2^{1} \; 2^{0} \; 2^{-1} \; 2^{-2}$$

Quanto vale?

$$5 + 1\cdot 2^{-1} + 1\cdot 2^{-2} = 5 + \frac{1}{2} + \frac{1}{4} = \frac{20+2+1}{4} = \frac{23}{4}$$

$$= 5{,}75$$

RAPPRESENTAZIONE IN VIRGOLA MOBILE (floating point)

ES.

$+6{,}25 \rightarrow$ $+625 \cdot 10^{-2}$ / $+0.625 \cdot 10^{1}$

- $+625$: numero intero
- 10^{-2}: numero intero relativo
- $+0.625$: mantissa
- 10^{1}: numero elevato ad esponente

NOTAZIONE SCIENTIFICA

Nei calcolatori:

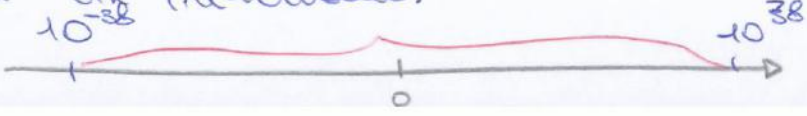
$$N = \pm M \cdot 2^{\pm E}$$

- N: numero relativo
- M: numero naturale
- base = 2 → perché siamo nel sistema binario

Ordine della memorizzazione nel calcolatore:

SEGNO - ESPONENTE - MANTISSA

Un numero in tale notazione utilizza di norma 32 bit pari ad un intervallo.



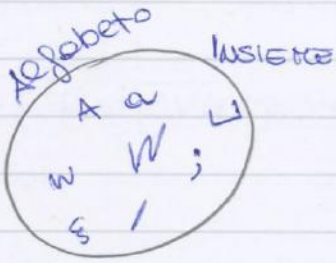
INFORMAZIONE NON NUMERICA (Caratteri)

ELABORAZIONE DELL'INFORMAZIONE NON NUMERICA

Noi rappresentiamo delle entità attraverso sequenze di bit.

00 01 10

La codifica dei caratteri avviene attraverso il codice standard ASCII, usato per lo scambio di informazione testuale.



$$N = \overbrace{int(\log_2 N)}^{\nu}$$

per sapere i bit

Codice ASCII

- Usa 8 bit per rappresentare:
  - 52 caratteri alfabetici (a...z, A...Z)
  - 10 cifre (0, ..., 9)
  - Segni di interpunzione (?, ; !...)
  - caratteri di controllo (es. quello per andare a capo)

↓

permette di fare 256 configurazioni

Lez 9 : INTRODUZIONE AL C

① 01011110 → ASCII

- numero intero assoluto
- numero intero relativo
- numero reale
- pezzo di un numero
- carattere di controllo

32 bit

S	E	M
1	8	24

②

```
  1 1 1
  0101   +5
  1011   -5
 ------
 -0000
```

overflow → segni operandi ≠ NO OV.

segni operandi = { segno risultato = NO OV.
" " ≠

STRUTTURA DI UN PROGRAMMA C

Parte dichiarativa e direttive

```
main ()
{
    Parte dichiarativa locale → scatole → int a;
                                          float b;
    Parte esecutiva
    return 0;
}
```

ISTRUZIONI DI STAMPA

PRINTF ()

printf (" ")

↓

Sequenza di caratteri che determina il formato di stampa di ognuno dei vari argomenti.

↓

(<formato>, <arg 1>, ... <argn>);

Come stampare — Cosa stampare

- #include <stdio.h>
- printf ("formato", valori);
- formato:
  - Testo libero (compresi spazi) → viene stampato letteralmente.

    ES. ("Marco, Torino")
  - Simbolo \n → va a capo
  - Simbolo %d → stampa un int
  - Simbolo %f → stampa un float

    ES. ("%d, %f", <nome 1>, <nome 2>)

ES

```
int a;
:
a = a + 2;
:
printf ("valore di a = %d", a);
```

ES 2

```
int a;
:
printf ("inteodusi' un numeeo = ");
scanf ("%d", &a);
```

↘ sequenza che mi permette di inteoodusee un numeeo.

ASSEGNAZIONI

< variabile > = < valoee >

↳ insieme di simboli

N.B non è un' uguaglianza.

- Assegnazione del valoee di una costante
  - i = 0;
  - a = 3.0;

  variab = numeeo
- Assegnazione del valoee di un' altea variabile
  - i = N;
  - b = a;

  variab = variab.
- Assegnazione del valoee di un' espressione
  - J = N - i;
  - b = a * 2 - 1;

  variab = variab + num.

Quesito

- Che operazione svolge il seguente frammento di programma?

```
a = 10;
b = 25;
a = b;
b = a;
```

a = b; → prende il valoee di b (25) e lo mette in a

b = a; → prende il valoee di a (che ora è 25) e lo mette in b.

a = b = 25

ES.

```
# include <stdio.h>
main()
{
   int a, b;
printf ("Dammi un numero intero (A): ");
 scanf ("%d", &a);
printf ("\n");
printf ("Dammi un numero intero (B): ");
 scanf ("%d", &b);
printf ("\n");
 printf ("A div B = %d\n", a/b);
 printf ("A mod B) = %d\n", a%b);
}
  return 0;
```

DIRETTIVE PRINCIPALI

- # include → "per eseguire il programma devi adoperare gli strumenti che trovi nella libreria <stdio.h>". (serve per definire le librerie; estensione)
- # define → serve per definire delle costanti

KLB

ES. (maiuscolo costanti, minuscolo variabili)

```
# define K 2
<nome> <valore>
```

FUNZIONI DI LIBRERIA — # include <math.h>

- funzioni algebriche
  - sqrt (radice quadrata)
- funzioni esponenziali
  - exp, log, log2, log10

ES $f(x, y, z) =$ _____

x	y	z	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

combinazioni possibili: 2 (num. variabili) ⇓ $2^3 = 8$

ESPRESSIONI BOOLEANE

AND → *

NOT → ' o −

X AND NOT (Y)

X * Y' oppure X * $\bar{Y}$

OPERATORE NOT

A	$\bar{A}$
Falso	Vero
Vero	Falso

Oggi è martedì pioe

NOT → Oggi non è martedì e non piove

OPERATORE AND

A	B	A X B
Falso 0	Falso 0	Falso 0
Falso 0	Vero 1	Falso 0
Vero 1	Falso 0	Falso 0
Vero 1	Vero 1	Vero 1

Oggi è martedì e pio
t V F

se una delle 2 frasi è falsa, la frase completa risulta falsa.

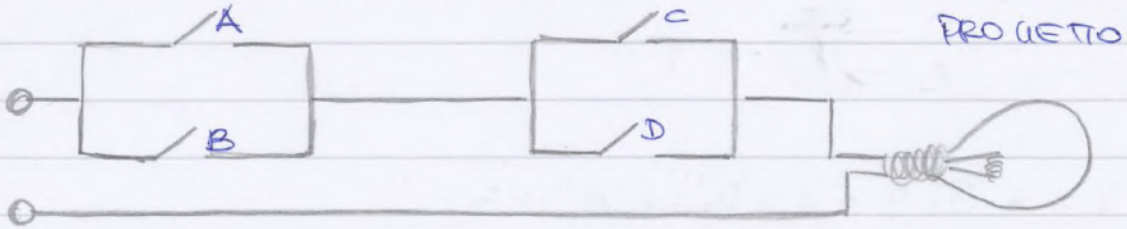
$\Rightarrow f(A,B,C) = (A \text{ or } B) \text{ and } C$

$(1 \text{ or } 0) \text{ and } 0$

$1 \text{ and } 0$

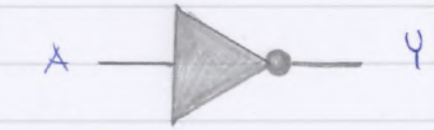
0

$f = (A \text{ or } B) \text{ And } (C \text{ or } D)$



Interessatori informatici sono i TRANSISTOR.

La Porta inv/not → funzione di inversione



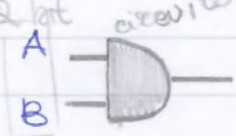
→ 0 → 1

→ 1 → 0

$Y = A'$

La Porta And

2 bit circuito che realizza la funzione and



$Y = A \times B$

I bit percorrono da sx a dx.

TEOREMI BASE

A And A' = falso = 0
A or A And B = A
A or A' and B = A or B
A or A' = vero = 1
A And (A or B) = A
A And (A' or B) = A and B
A or 1 = 1
A * 1 = A
A * 0 = A
A or falsa 0 = A

$x + \bar{y} + x y + \dots$

Quando una funzione è complessa la semplifico.

$x(1+y) + \bar{y}$

$x \cdot 1$

x

$(x+\bar{y})\,\bar{x}\,y + \bar{z}$

$x \cdot \bar{x} \cdot y$

$0 \cdot y$

0

---

f_1 f_2

$A + (B \times C) = (A+B) \times (A+C)$ → funzioni di 3 variabili A, B, C.

Le due funz. sono equivalenti?
Se due funz. sono ≡ la loro tabella di verità è =.

$f_1 = A + (B \times C)$

Calcolo la tab. della verità

A	B	C	f_1
0	0	0	0
0	0	1	
0	1	0	
0	1	1	1
1	0	0	
⋮	⋮	⋮	

sostituisco

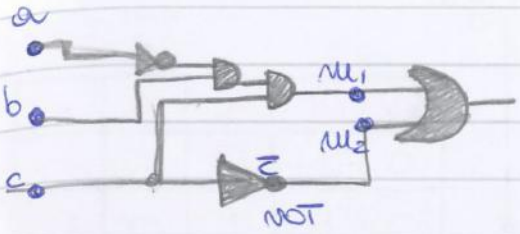
Faccio la tab. della verità della f_2 e se è = alla tab. della verità di f_1 le due funz. sono equivalenti.

ES. $F = \bar{a}bc + \bar{c}$ (M_1, M_2; + = OR)

→ a, b, c → Sono i bit che entrano nel mio circuito

come costruisco il circuito?

M_1 e M_2 sono tenuti insieme da OR.



$M_2 = \bar{c}$

$M_1 = \underbrace{\bar{a}bc}_{a \times c}$ → si vuole l'AND tra 3 ingressi

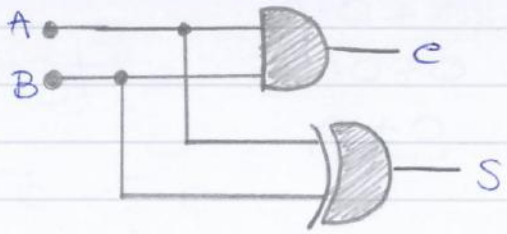
SOMMATORE

f_1 = somma → S

f_2 = carry → C

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Circuito che fa la somma di 2 bit:



ES. CASA

$f_1 = \overline{x}\,\overline{y} + \overline{z}\,x$

$f_2 = \overline{(x + \overline{y})} + xy$

Sono equivalenti?

x	y	z	f_1	f_2
0	0	0	1	0
0	0	1	1	0
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

1) $\overline{x}\,\overline{y} + \overline{z}\,x$

$\overline{0 \cdot 0} + \overline{0} \cdot 0$

$\overline{0} + 1 \cdot 0$

$1 + 0$

1

2) $\overline{(x + \overline{y})} + xy$

$\overline{(0 + \overline{0})} + 0 \cdot 0$

$\overline{(0 + 1)} + 0 \cdot 0$

$\overline{1} + 0$

$0 + 0$

0

2) $\overline{0 \cdot 0} + \overline{1} \cdot 0$

$\overline{0} + 0 \cdot 0$

$1 + 0$

1

$\overline{(0 + \overline{0})} + 0 \cdot 0$

$\overline{(0 + 1)} + 0$

$\overline{1} + 0$

$0 + 0$

0

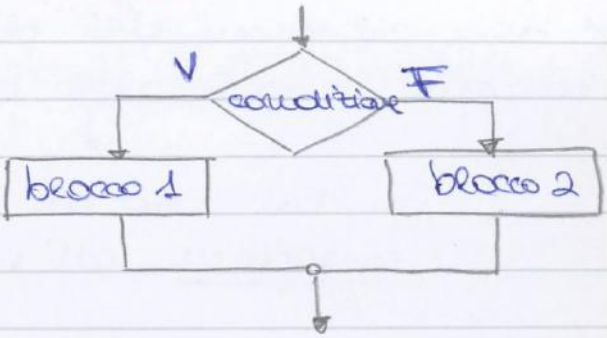
⇓

Le due funzioni non sono equivalenti

ISTRUZIONI IF - THEN - ELSE

- SINTASSI:

```
if (<condizione>)
  {<blocco 1>}
else
  {<blocco 2>}
```



la condizione può contenere

operatori relazionali:
- confrontano due entità a e b attraverso criteri matematici. Si dicono se

a == b (simbolo dell'uguaglianza (= è l'assegnatore))

a > b
a < b
a >= b
a <= b
a != b

operatori booleani:
- AND (&&)
- OR (||)

ES. if ((condiz)^1 AND (condiz.)^2)

ES Leggere due valori A e B, calcolarne la differenza in valore assoluto D = |A - B| e stamparne il risultato.

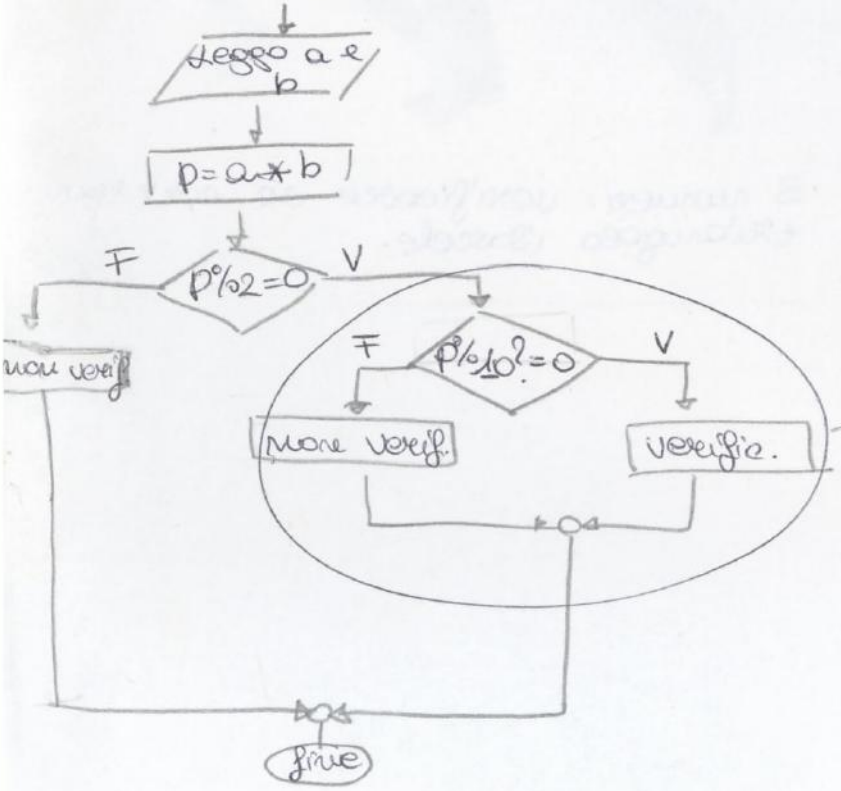
```
main ()
{ int A,B,D;
  scanf ("%d %d", &A, &B);
  if (A>B)
    {D= A-B;}
  else (B>A)
    {D=B-A;}
  Printf ("%d\n", D);
```

```
int a,b, p;
printf ("1° numero = ");
scanf ("%d", &a);
printf ("2 numero = ");
scanf (" %d", &b);
p= a*b;
if(p/2=0 ) → l'operatore del resto è % → (p%2==0)
 {printf ("
 p è pari")};
else
 {printf ("
 p è dispari")};
return 0;
```

ES 2

IF ANNIDATO

a b → p è pari e contemporaneamente è divisibile per 10.



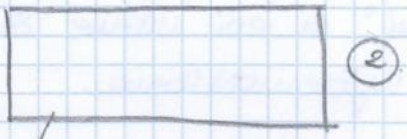

```
printf
scanf
printf
scanf } = prima
p=a*b;
x= p%2
if (x==0)
{ x = p%10;
  if (x==0)
   {printf("verif");
   else
   {printf("non verif");

}
else
 {printf("non verif");}
return 0;
```

Un' immagine è codificata su un file.

① 

- L'immagine viene rappresentata da un insieme di pixel.
- Poi viene visualizzata su uno schermo.



Ma non è detto che il num. di pixel che abbiamo in ① sia uguale al num. di pixel che abbiamo in ②.
Es. Film a bassa definizione e TV ad alta definizione.

⇩

Qui si vede meglio!

I pixel di schermo si chiamano DOT.
Quello che conta è dunque l'informazione che abbiamo sul file.

RAPPRESENTAZIONE VETTORIALE

L'immagine non è presa dall'esterno ma è costruita geometricamente all'interno del calcolatore.

La codifica in bit ha il vantaggio di comprimere l'informazione in maniera ageyole (data compression)

Le tecniche di compressione si dividono in 2 grandi categorie:

- senza perdita (lossless): l'operazione di decompressione permette pertanto di recuperare tutti gli elementi iniziali.
- con perdita (lossy) i dati compressi non contengono tutta l'informazione iniziale.
  L'operazione di decompressione non permette pertanto di recuperare tutti gli elementi iniziali.

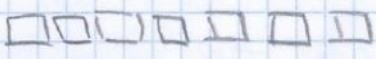
FORMATI RASTER sono normalmente di questo tipo:
BMP e GIF e PNG

JPEG
- è il più adoperato nel mondo fotografico per compressione con perdita.
- è un algoritmo estremamente raffinato.

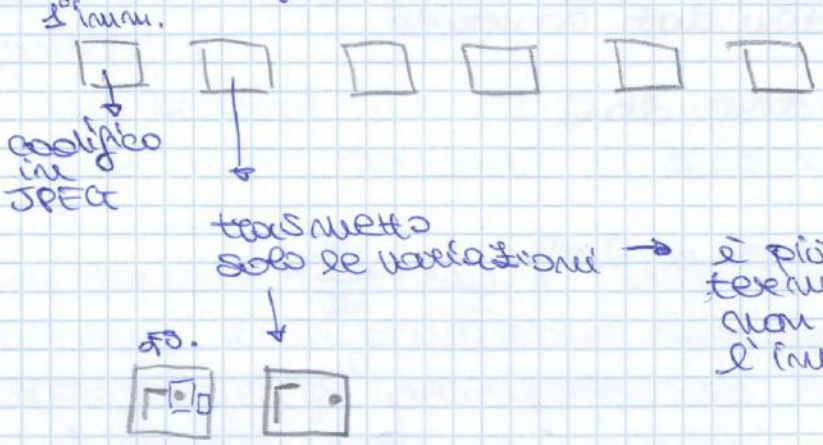
IMMAGINI IN MOVIMENTO
Il movimento è simulato dalla sequenza di fotogrammi.

- FILMATO = sequenza di immagini (ciascuna chiamata FOTOGRAMMA o FRAME).
- minimo 16 frame/s affinché l'occhio umano non percepisca i singoli fotogrammi.

25 frame/s = 25 Hz

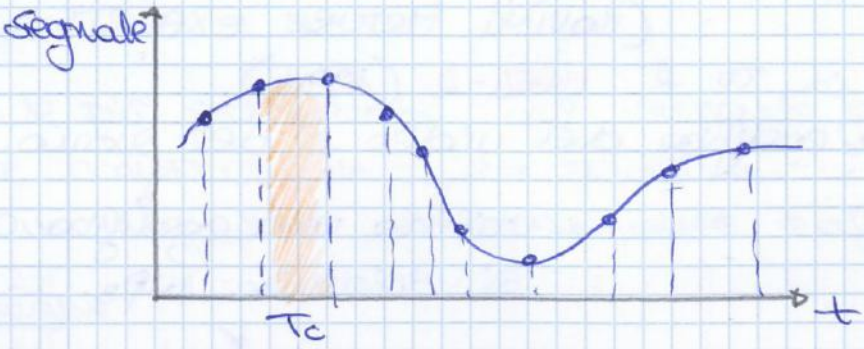


come comprimerò?
Memorizzando su un DVD riesco a comprimere più frames e a farci stare più immagini → film più lunghi



Dopo un po' di frames, quando tutte le variazioni sono troppe, ritrasmetto l'immagine codificata.

La traduzione in bit avviene attraverso un meccanismo che viene chiamato CAMPIONAMENTO



Ad intervalli regolari ne definisce il valore di quell'andamento.

Frequenza di campionamento

$$f_c = \frac{1}{T_c} \qquad \text{udm} = \text{Hertz}$$

Più gli intervalli sono piccoli, maggiore è la precisione nella rappresentazione del segnale.

FORMATO CD AUDIO

- campionamento = 44 kHz
  è espresso in 16 bit
- deve avere 2 canali.

MP3 → prende un suono.
- Calcola le componenti di quel suono.
- Delle frequenze prende solo le più rilevanti.

FORMATO MIDI
- codifica uno spartito musicale
- solo musica, non voce umana
- per riprodurre i suoni richiede "campioni" dei vari strumenti
- molto efficiente
- usato da videogiochi e siti web

dez 13:

ESERCIZI DI PROGRAMMAZIONE

① Supponiamo di voler realizzare un programma che calcola il prezzo di un'automobile.
- prezzo di base 7'000 €
- aumentato dagli optionals (500 €)
- prezzo è diminuito della fedeltà al marchio per ogni mese (sconto 80 €)
- il prezzo comunque non può essere inferiore a 6'000 €.

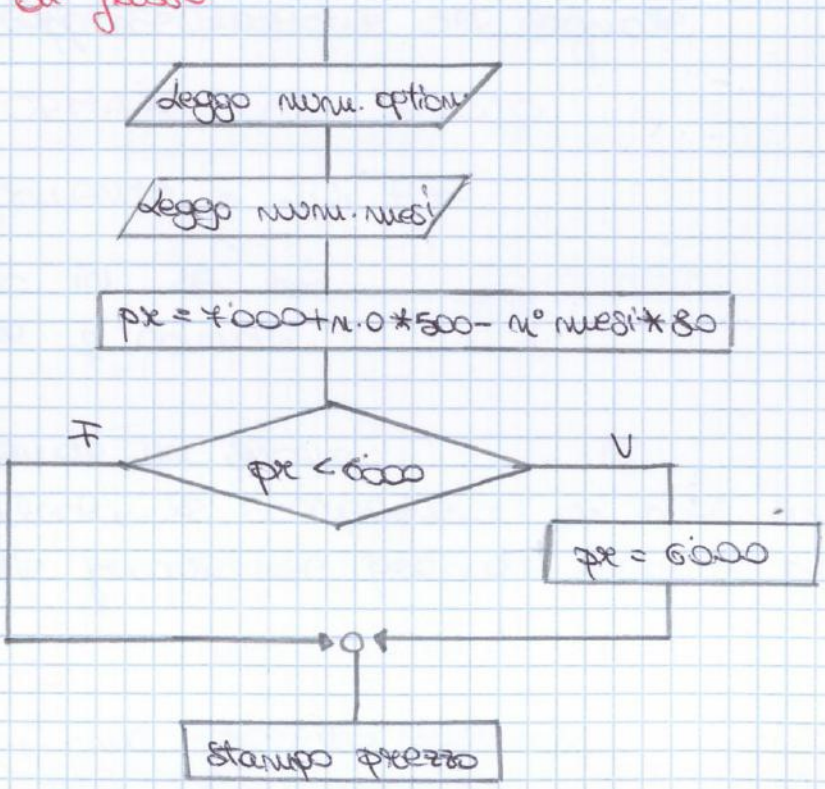
DATI:

Costanti sono: 7'000 €, 80 €, 500 €

Variabili: n° optionals, n° mesi di fedeltà, prezzo finale

Incognite: prezzo

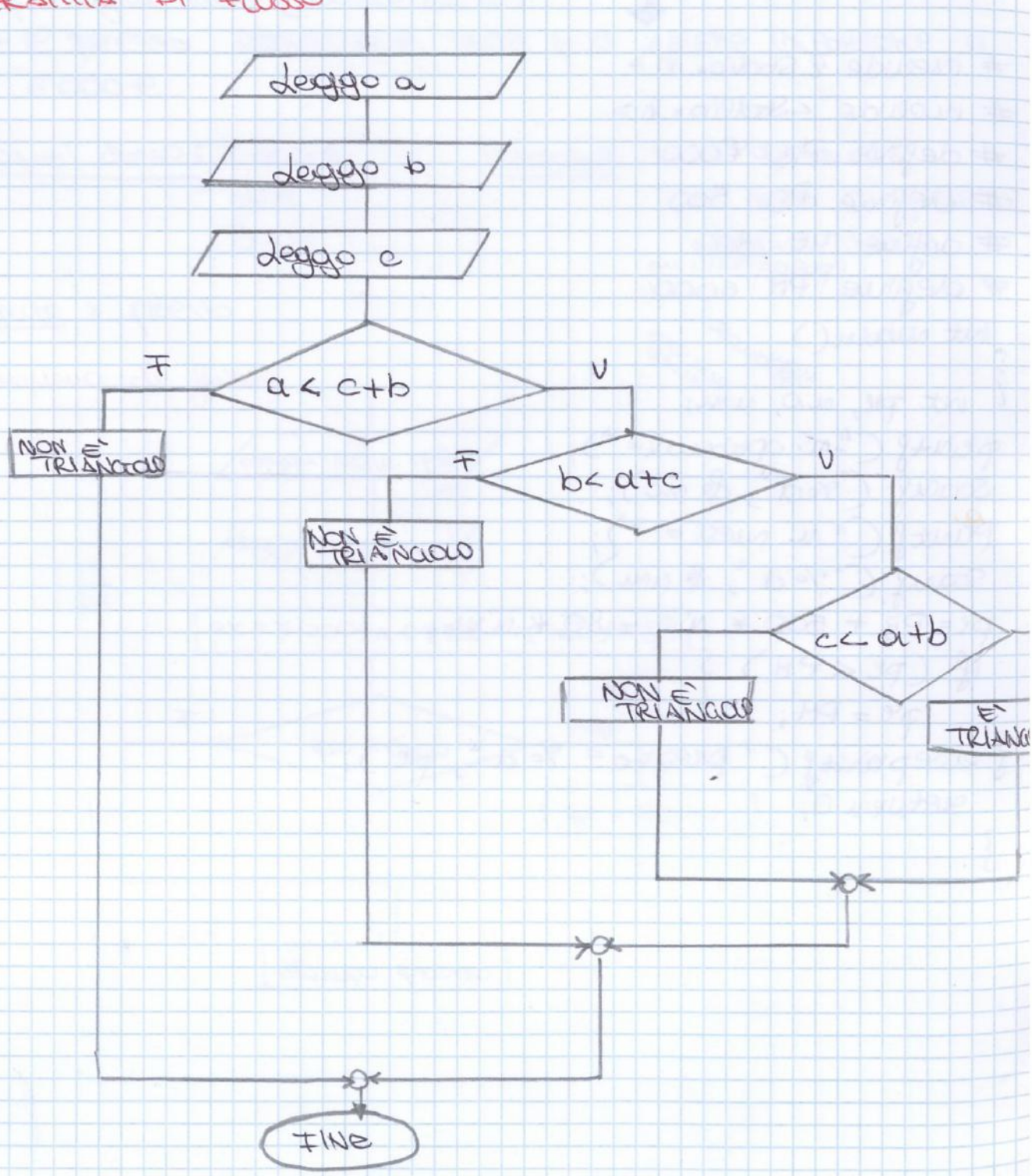
Diagramma di flusso



② Dati 3 numeri interi verificare se sono i lati di un triangolo.

Avremo 3 variabili: a, b, c
Condizione affinché 3 numeri siano i lati di un triangolo: ciascun lato deve essere < della somma degli altri due.

DIAGRAMMA DI FLUSSO



```
printf ("TRIANGOLO");
else
printf ("no triangolo");
```

ISTRUZIONE SWITCH

Si usa quando si hanno delle scelte multiple.

Sintassi:

```
switch (<espressione>)
{
 case <costante 1> :
    <blocco 1>
 break;
case <costante2> :
  <blocco 2>
 break;
...
 default :
    <blocco default>
}
```

Es. x > 3 (NO) → operatore relazionale
x * x + 3 (SI)

Dove

<espressione> : Espressione a valore numerico
<blocco 1>, <blocco 2> : sequenza di istruzioni

Es.

```
int x;
...
switch (x) {
 case 1:
  printf ("sono nel caso 1\n");
 break;
 case 2:
   printf ("sono nel caso 2\n");
 break;
 default :
   printf ("Né caso 1 né caso 2 \n");
 break;
}
```

- Sintassi:

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

dove:

<condizione>: condiz booleana
<blocco>: sequenza di istruzioni

↓

Ripeti <blocco> finché <condizione> è vera.

Es. leggere un valore N, calcolare la somma S dei primi N numeri interi e stamparla.

```
#include <stdio.h>
main () {
 int N, i, S;
 i = 1; S = 0;
 scanf ("%d", &N);
 while (i ≤ N) {
     S = S + i;
      i ++;
 }
  printf ("Somma = %d\n", S);
}
```

$i = i + 1 \iff i++$

$i = i - 1 \iff i--$

PROGRAMMA:

```
#include <stdio.h>
#include <stdlib.h>
int main ()
{ int S, i, n;
  float media;
  i=0;
  S=0;
printf ("dato = ");
  scanf ("%d", &n);
while (n>=0)
    S= S+n;
     i= i+1;
printf ("dato = ");
  scanf ("%d", &n);
}
media = S/i;
media = (float) S/i;
  return 0;
}
```

```
if (n<0)
printf ("no buono!");
else{
}
printf ("media = %f", media); }
```

numero degli elementi che ho inserito

scrittura errata

scrittura corretta

Programmia:

```
#include <stdio.h>
#include <stdlib.h>
int main ()
{int s, i, n, nmax;
 float me;
  printf ("n. elementi = ");
 scanf ("%d", &n max);
  s=0;
  i=0;
do { printf ("elem= ");
     scanf ("%d, &n);
     s=s+n;
     i=i+1;
while ( (n!=0) && (i<nmax));
  if (n=0)
     i=i-1; → tolgo lo 0
  me = (float) s/i; num. elementi - 0
}else
  me = (float) s/nmax;
  printf ("media = %f", me);
return 0;
}
```

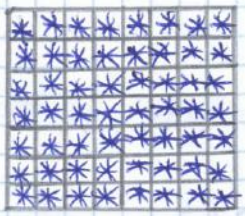
Programma:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  int a,b,i, imax, pmax,p;
   pmax=0;
    i=0;
   printf("1° elemento = ");
    scanf("%d", &a);
   printf("2° elemento = ");
    scanf("%d", &b);
while (b!=0) {
        i+1;
     p=a*b;
  if (p<0)
        p= -p;
  if (p>pmax) {
        pmax=p;
        imax=i;
  }
     a=b;
   printf("2° elem = ");
    scanf("%d, &b);
  }
    printf("max=%d - indice = %d", pmax, imax);
return 0;
}
```

ho la 1ª coppia devo fare il while

il 2° elem. della coppia diventa il 1° della nuova coppia



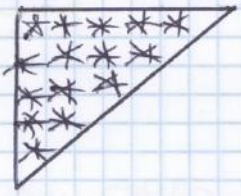
Per costruire la figura:

```
i=0
while (i<n)
{
    j=0;
    while (j<n)
    {
    printf ("*");
      j=j+1;
    }
    printf ("\n");
    i=i+1;
}
```

→ mi stampa la 1ª riga

→ dice quante righe stampare.

④ Stampare un triangolo rettangolo di lato N ("*")



STRATEGIA:
Stampa una riga → ripetilo N volte


```
i=0
while (i<N) {
   stampa riga
}
```

```
j=0
while (j<K) {
printf("*")
j+1;
}
```

stampa K asterischi