NUMERO: 2375A ANNO: 2018

A P P U N T I

STUDENTE: Zarrelli Mattia

MATERIA: Zarrelli Mattia - Informatica - Prof. Mezzalama

INFORMATICA

Ambienti di sviluppo x programmazione

- CODEBLOCKS (cross-platform) → istruzioni caricate più avanti
- http://www.codeblocks.org

Esame: { 3 domande di Teoria ← necessaria la sufficienza
Scrittura di un programma in C

La società digitale

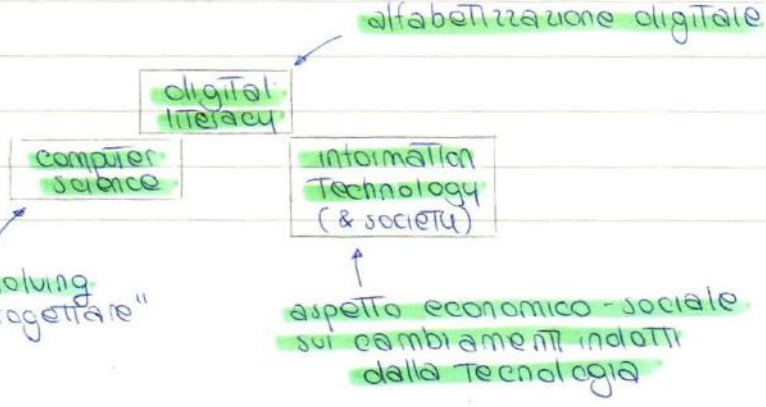
Società post-industriale caratteriz. dalla gestione dell'informazione e dell'energia. Nei secoli passati della società industriale ci sono stati grandi cambiamenti Tra fine 1700 e inizio 1800 e Tra 1800-1900 e 1900-2000. Cambiamenti determinati dall'introduzione di elementi Tecnologici.

- 1700-1800 motore a vapore (Treni che cambiano economia, Trasporti ad es.)
- 1800-1900 3 innovazioni { · energia elettrica (→ urbanizzazione)
  · motore a scoppio (→ ci si sposta verso mezzi individuali, movimento a proprio piacimento)
  · chimica, nuovi materiali

  → Rivoluzioni essenzialmente energetiche → allontanarsi dalla fatica fisica dell'uomo.
- 1900-2000 introduzione delle Tecnologie digitali → gestione dell'informazione
  Se prima le macchine levavano fatica fisica ora le nuove Tecnologie assistono la mente umana (sono più efficienti e più pericolose).

Ingegnere deve essere cosciente dell'impatto che ha ciò che progetta, ha una responsabilità anche sociale.

I saperi dell'informatica



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 3 di 134

Big data → quanto Big? → Informazione creata e scambiata ogni anno è di (6 ZETTABYTE ($6 \cdot 40^{21}$ BYTE)
~ 323 miliardi di libri come la divina commedia

IoT = "Internet of Things"
Internet nato per comunicare, da molte cose sono informatizzate e devono comunicare, collegare tanti dispositivi ⇒ Internet è ora anche una rete tra le cose.

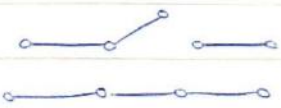
Nuovi comandamenti → Un processore (intelligenza della macchina) in ogni oggetto o essere
Un indirizzo IP (indirizzo per collegarsi) in ogni oggetto o essere
⇒ Intelligenza + Connettività

Anche la Televisione → digitale → satellitare

Digitale vs Analogico
Perché codificare in digitale è meglio?
- Qualità del dato → informazione nel Tempo non degrada (come avveniva con le cassette ad esempio)
- Efficienza della Trasmissione → compressione dei dati = ridurre quantità di simboli che ci permette di descriverla
  - meno spazio per memorizzare
  - Trasmetto soltanto info essenziale

Linguaggio / alfabeto utilizzato
→ calcolatore usa solo (2) simboli → + SEMPLICE
il nome di questi due simboli è il (BIT) = "Binary digIT" = cifra binaria

Ma cos'è?
L'informazione è gestita in interruttori → aperto = (0)
chiuso = (1)



Interruttore chiamato TRANSISTOR

Ho bisogno di rappresentare più informazione e devo quindi raggruppare più BIT
⇒ gruppo di 8 bit = BYTE
- bit = b
- byte = B

es. $1GB = 1 \cdot 1024 \cdot 1024 \cdot 1024 = 1'073'741'824$ byte

→ sono un po' di più di un miliardo

(3) Trasportare vuol dire quanta informaz. passa da un punto a un altro in un periodo di Tempo → bit al secondo = bps

Informazione che passa da un punto a un altro in un ΔT

- Kbps = 1000 bit al secondo
- Mbps = 10^6 bit al secondo

Modem/Router a 2MB = 16Mbps

⇒ collegamento ADSL = "asimmetric digital subscription line"

↳ perché un canale di comunicazione Tipico la portata di informazione ricevuta è molto maggiore di quella data a un server

ADSL ⇒ Velocità download > Velocità upload

cellulari Trasmettono voce e dati → es. 4G/LTE = 100Mbps

3G = 40 Mbps

Trasmissione dipende anche però dalla rete

→ capacità cellulare + capacità rete

es. filmato di 10 min → quanto ci vuole x usufruirne?

costituito da una serie di "frame" (immagini)

Sensazione del movimento almeno 25 frame al secondo

Un'immagine è composta da pixel, ogni pixel con 8 bit

grandezza del formato immagine 720 x 576

8 bit per 1 pixel → 720x576 pixel in una immagine → 25 immagini al secondo → 83 Mbps

ogni immagine = $720 \cdot 756 \cdot 8 = 3'317'760$ bit

25 frame al secondo = 82'944'000 bit al secondo

⇒ mi servono circa 83Mbps → lo vedo con il 4G

Si può volendo comprimere con codifiche adatte, meno bit necessari

Meno pixel, meno colori = $25 \cdot (720/2) \cdot (576/2) \cdot 8/2 = 10'368'000$ bps

→ lo vedo con 3G e 4G

⇒ fibre ottiche = da 10Tbps (in una sola fibra possono essere veicolate Tutte le Telefonate di una nazione)

Tipi di computer

(1) "di uso generale" → general purpose (il pc comune)
(2) "dedicati" → special purpose (il cellulare)

La differenza è che nel primo posso cambiare dinamicamente programma mentre nel secondo il programma è fisso.
Il mondo è pieno di computer special-purpose, nascosti negli oggetti.

Calcolatori di un'azienda stanno solitamente nella "server farm"
I calcolatori più potenti = mainframe o super pc

Rapporto computer-energia
Devono consumare poco, per durare nel tempo

Super pc = 20 MW
A casa mediamente = 3 kW di contratto
⇒ super pc consuma come 10.000 case

PROBLEM POSING & SOLVING

= Analizzare un problema e Risolverlo — COMPUTATIONAL THINKING
processo concettuale del PP&S è:
1) Formulazione del problema e capirlo
2) Esprimere una soluzione (codificarla)
3) Verificare la soluzione

Ingegnere → Risolvere problemi posti (in TUTTI i campi)

PROGETTARE = Problema → soluzione → soluzione formale → Realizzazione
È un processo per approssimazioni successive in cui si può anche Tornare indietro.

Algoritmo = descrizione precisa (formale) di una sequenza finita di azioni che devono essere eseguite per giungere alla soluzione di un problema.

= le azioni che devo eseguire per risolvere un problema indicate con precisione

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 9 di 134

Prima della scrittura del programma si dichiarano le librerie da utilizzare.

Diagrammi di flusso e strutture dati — lezione 3

Flow-chart = composti da simboli grafici PREDEFINITI → Blocchi e ARCHI orientati

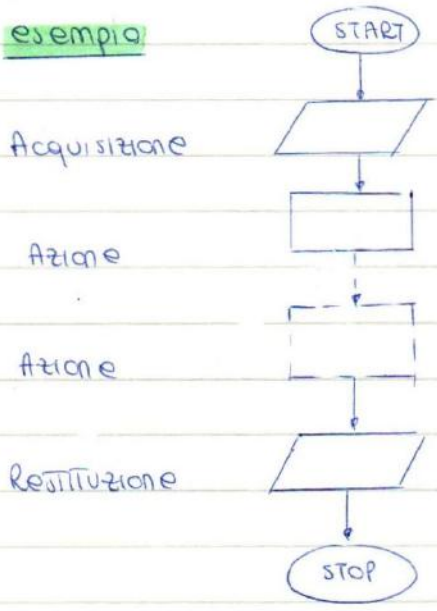
1. Blocchi inizio/fine

2. Blocchi di azione

= Prendere INFO dall'ext o produrre INFO per l'ext (acquisizione o restituzione INFO)

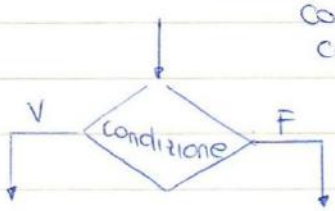
Scrivo qualcosa dentro, un comando

esempio

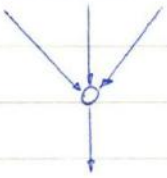


3. Blocco di DECISIONE

Condizione di carattere booleano



4. Blocco di connessione



Convergenza in un unico punto.

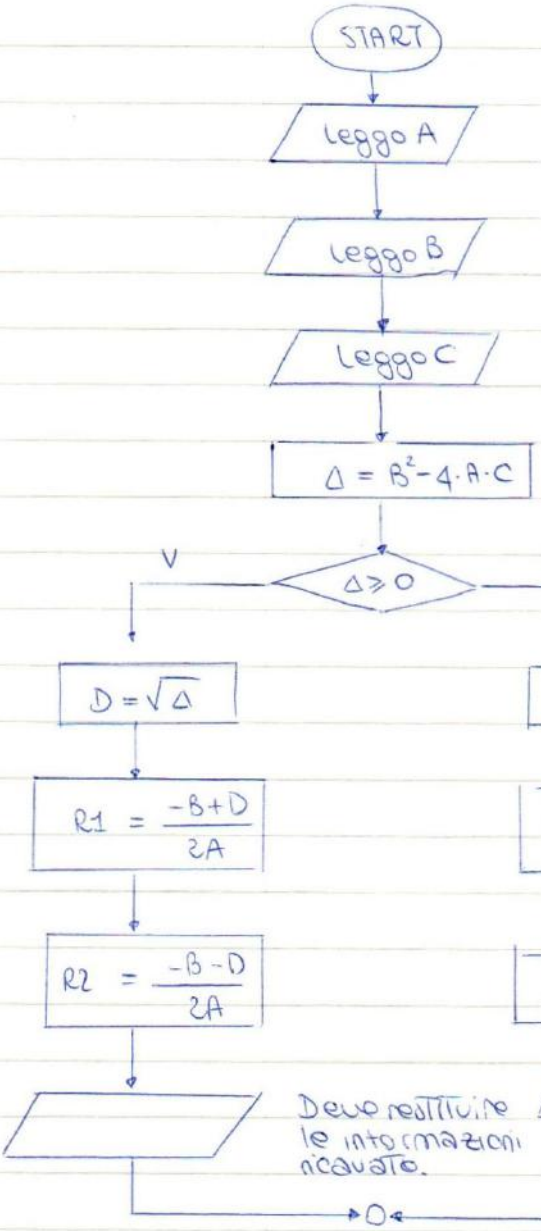
Loop = "giro" = ripetizione che permette di rieseguire le condizioni considerate.

Esempio: Calcolare le radici di $ax^2 + bx + c = 0$

$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

= Radici

Prima lo facciamo senza le condizioni limite.



Soluzioni complesse

Devo tenere conto delle condizioni al contorno:

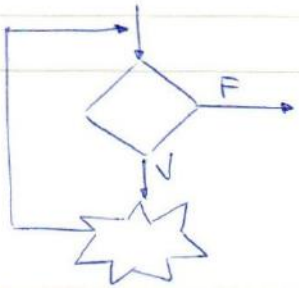
- $A = 0$ ⇒ programma che si pianta (eq. primo grado)
- $B = 0$ ⇒ funziona
- $A, B, C = 0$ ⇒ NON funziona

con $A = 0$ ⇒ $x = -C/B$ con $B \neq 0$

Diagrammi di flusso rispecchiano le condizioni mentali di ragionamento "IF-THEN-ELSE"

- IF-THEN = se si fa una cosa, se no non fare nulla
- WHILE-DO = Ripeti un'azione tante volte finché non succede una data situazione

- DO-WHILE



V1 

V2 

Vedere se V2 è dentro V1

Controllare primo elemento con primo elemento finché non combaciano. Quando combaciano si controlla secondo elemento. Verifica da fare fino a V1-V2 poiché devono esserci abbastanza caselle per sviluppare l'algoritmo.

Esempio: Area 2000 x 1000 suddivisa in tane 100x100 dove effettuo perforazioni che danno valore 1 o 0 in base alla possibilità di presenza di gas.

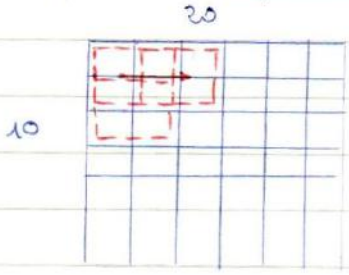
(1) Quante aree con indice >= 7

(2) Aree 200 x 200 degna di essere perforata se somma delle aree 100 x 100 in essa contenuta è >= 34

Matrice 20x10 (20 colonne e 10 righe)

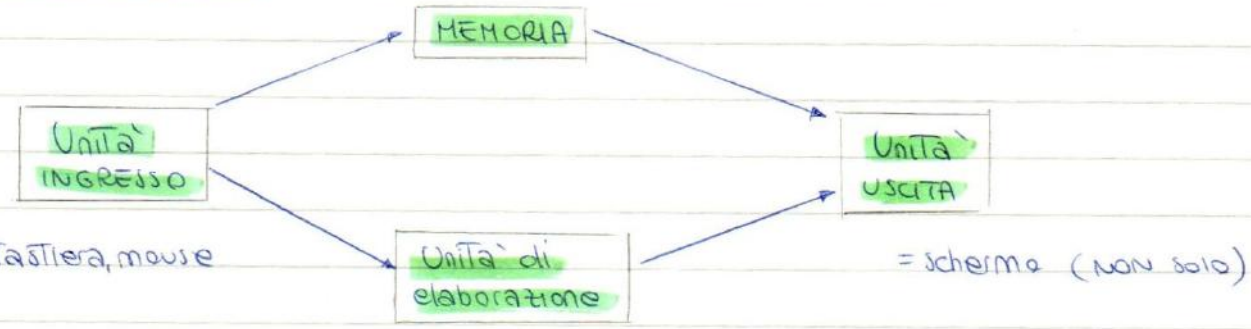
(1) Indici da 0 a 19 e da 0 a 9 e vedere valori

(2) Devo prendere 4 quadrati per volta e vedere la somma



La struttura del calcolatore (CPU, bus, memoria) - lezione 4 -

Blocchi fondamentali



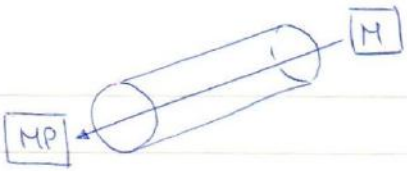
= Tastiera, mouse

= schermo (NON solo)

= CUORE = CPU

= central processing unit

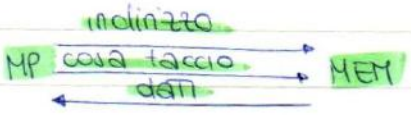
= quella che esegue il programma ricevendo informazioni



BUS come Tubo
Tra CPU e memoria.

↳ caratterizzato da 1) FREQUENZA = n° dati trasportati al secondo (es. 100 MHz) → Bus compatibile con il MP (simile velocità)
↑ "quante volte trasporta"
2) AMPIEZZA = n° di bit del singolo dato

3 classi di informazione del BUS



- ABUS = adress bus
- CBUS = control bus
- DBUS = informazione dei dati

→ Sono proprio fili separati

con A bus specifico gli indirizzi → devo "contare" fino al numero che mi serve
con Dbus specifico la quantità che trasporto → Se trasporto 1 byte = 8 bit = 8 fili

Nomi dei BUS ad esempio PCI o PCI express (PCIe)

Esempio: Ho 10 fili (BUS), quanto è grossa la memoria?
Posso specificare 2^{10} indirizzi = capacità della memoria =
$2^{10} = 1024$ indirizzi = quanta memoria posso avere

Quanto è grossa?
Devo capire quanto è larga (quanti bit contiene) ogni cella di memoria
Il Dbus deve essere grande quanto la cella per trasportare TUTTO il contenuto

$$\text{Max MEM} = 2^{|\text{Abus}|} \cdot |\text{Dbus}| \text{ bit}$$

bit di ogni Dbus = = dimensione cella di memoria

DISPOSITIVI PERIFERICI

Devo prendere INFO dalla memoria → mandarle alla periferica attraverso MP
MP manda prima all'interfaccia → periferica

Porte attraverso le quali trasmette (USB)

Chip = circuito integrato che contiene miliardi di transistor
↳ costruisco il calcolatore

Smontando un PC ... piastra elettronica (BOARD)
con connettori = connessioni meccaniche dove inserisco dispositivi
↳ per microprocessori, schede grafiche ecc.
(se non è saldato)

Memoria =
= insieme di CASSETTI. Caratterizzata da 3 elementi

ogni locazione di memoria ha stessa capacità

(1) Indirizzamento (ogni cella con una etichetta) → veicolato da A bus
(2) Parallelismo (quanti bit in // all'interno del cassetto)
(3) Accesso (velocità con cui leggo/scrivo un dato nella memoria)
(o Tempo di accesso)
- circa 60 ns ad es. -

Memoria interna { RAM — letta e scritta da µP
ROM — letta da µP

memoria allo stato solido → NON VOLATILE
{ ROM contiene info anche se NON alimentata → TUTTI i PC ne hanno almeno 1
RAM non contiene se non alimentata

All'accensione il µP deve andare a recuperare le prime istruzioni dalla ROM. La RAM parte con informazioni del disco (Windows) presa dal µP.

Memoria esterna { Magnetica
Ottica
↓
Memorie più lente di quella interna (10^{-3} s)

MEMORIA CENTRALE

Sistema operativo	RAM
Programmi	RAM
Memoria video	RAM video → Informazione che deve essere proiettata sullo schermo
Programma d'avvio	ROM

(boot program = BIOS)

{ ROM ~ 128 kB
RAM ~ 4GB
RAM video ~ 1GB

Nei calcolatori dedicati anche i programmi stanno in ROM (es. lettore MP3)

Esempio: i7 da 4 a 6 core; Architettura a 64 bit (larghezza celle memoria)
Transistor 20 nm; Cache da 64k per L1 a 8MB per L3;
frequenza fino a' 3.5 GHz.

LINGUAGGIO C • Sviluppato Tra il '69 e il '73 per scrivere un sistema operativo (UNIX)
- Linguaggio ad Alto livello, indipendenti dall' hardware.
  ↳ linguaggio più diffuso

Elementi del linguaggio: LESSICO, SINTASSI, SEMANTICA
↓ parole frase significato
dipendono dal programma Dipendono da noi
↓
Calcolatore verifica queste due parti

- PAROLE CHIAVE
- IDENTIFICATORI
- ISTRUZIONI

DATI

Come chiamo un dato? Come lo metto in memoria?

Dato ≃ scatola che deve avere un NOME (identificatore) e deve avere un TIPO (interpretazione) in modo da dimensionarla nel modo giusto.

3 Tipi: carattere, scatola da 1 byte = CHAR
interi = INT + DOUBLE (reali con > precisione (doppia))
numeri con virgola, reali = FLOAT

All inizio del programma devo specificare quali scatole voglio usare.
Così vado in memoria e costruisco le scatole che mi servono.
→ Posso dare mettere un valore predefinito: int pippo = 17

Posso avere due tipi di scatole → VARIABILI = posso cambiare contenuto
→ COSTANTI = contenuto non più variabile

const int 17
manca nome

Programma in due parti:
(1) DICHIARATIVA = definisco i dati (le scatole)
(2) ESECUTIVA = scrivo le istruzioni che manipolano i dati

Inizio = main ()
Fine = return 0
parentesi graffe includono porzioni di codice, anche inizio e fine programma.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 21 di 134

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

esempio:

```
main ()
{
    float a;
    a = 3,2;
    a = a + 0,5;
    printf ("%f", a);
    return 0;
}
```

Se ho più variabili di diverso tipo metto tutti di conversione in ordine

```
printf ("%d %f %c \n", x, z, c)
```

fattore che dà il comando di andare a capo dopo aver stampato.

Per stampare messaggio `printf ("benvenuto");`

Istruzione scanf

(introdurre dati da tastiera) → prende valore che immettiamo e lo mette nella scatola con quel nome

Sintassi come printf → variabile preceduta dalla (&)

```
scanf ("%d", &a);
```

Esempio: moltiplicare numero per 2

```
# include <stdio.h>
main ()
{   float n;
    scanf ("%f", &n);
    n = n*2;
    printf ("%f", n);
    return 0;
}
```

Esempio: legge 2 numeri e moltiplica

```
# include <stdio.h>
main ()
{   int a;
    int b;
    int c;
    scanf ("%d", &a);
    scanf ("%d", &b);
    c = a * b;
    printf ("%d", c);
    return 0;
}
```

Esempio: leggere un numero, moltiplicare per 13, stamparlo

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

int main ()
{   int a ;          → meglio semetto prima printf ("numero= ");
    scanf ("%d",&a) ;
    a = a * 13 ;     → oppure printf ("valore di a = %d", a)
    printf ("%d", a) ;
    return 0 ;
}
```

calcolare la radice $R = sqrt(a)$ ⟹ warning che manca libreria
↳ #include <math.h>
mancherebbe float R; anche.

Codifica dell'informazione

(Rappresentazione numeri e operazioni in binario)

codice binario → unici elementi sono 0 e 1

Con N bit posso avere 2^N combinazioni
↳ con 3 bit posso rappresentare 2^3 combinazioni diverse

Al contrario: se devo codificare k oggetti, quanti bit mi servono?

$$n \geq \log_2 k$$
$$n = int(\log_2 k) \quad \text{con } n = \text{intero}$$

esempio: rappresentare numeri da 0 a 17

$n \geq \log_2 17 \rightarrow n = int(\log_2 17) = (5)$ $(2^4 = 16)$
↳ 32 possibili sequenze, ne prendo 17.

Come contiamo?

sistema posizionale ——— base

(10) = anche al numero di cifre utilizzabili $(0, 1, \ldots, 9)$

esempio:

base 4 $\{0, 1, 2, 3\}$

231 in base 4 $= 231_{(4)}$ $\qquad 251_{(4)}$ = non leggibile

$2 \cdot 4^2 + 3 \cdot 4^1 + 1 \cdot 4^0 = 45$ in base 10 $= 45_{(10)}$

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 25 di 134

- lezione 7 -

Inizio laboratori

Sistema esadecimale

Raggruppo a 4 a 4 le cifre, sistema con 16 simboli → {0,1,...,9,A,B,C,D,E,F}

1011 1001

(11) B 9 → Scrivo così solo 2 cifre

1FA → 0001 1111 1010

Numeri relativi (interi relativi)

± 33 → Nel calcolatore? → Modulo e segno

Complemento a due (unica nei calcolatori)

Modulo e segno

Il segno deve essere un bit → bit + a sinistra rappresenta il segno

segno modulo

0 = positivo
1 = negativo

Soluzione non adoperata

perché nei calcoli dà problemi

ad esempio: (-2) + (+2) = 0

1	0	1	0
0	0	1	0
1	1	0	0

= (-4)

perché il sistema non è più posizionale e non posso applicare le stesse regole.

Complemento a due

Sistema posizionale in cui TUTTI sono positivi mentre quello più a sinistra è negativo

esempio: -2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

↳ 101101

$-32 + 8 + 4 + 1 = -19$

Se c'è 0 all'inizio → Numero positivo

↳ Il primo bit definisce nuovamente il segno ma in modo diverso.

(-5 / +5)

1011 +
0101
0000 Giusto

Da numero decimale a complemento a 2?

- Se positivo metto lo 0 iniziale e scrivo normalmente 4 → 0100
- Se negativo?

-4? Rappresento +4, poi inverto ogni bit e sommo 1

↳ 0100 → 1011 → 0001 + → 1100 (-4)

↳ 1011

= operazioni di complementazione (trovano l'opposto) partendo anche dal negativo

con 4 bit intervallo da 1000 (-8) a 0111 (7)

Rappresento intervallo molto più grande con virgola mobile rispetto a complem a 2 (compl.)

es. 32 bit → da -2 miliardi a +2 miliardi (-10^9 / 10^9)

-10^{38} $+10^{38}$ ← virgola mobile

int a; codifica in complemento a 2 (intero relativo)

float a; codifica in virgola mobile

Come rappresento 1/3? Non è una sequenza di cifre sensibile 0,3̄.
oppure π, e ...
⟹ Numeri intrinsecamente approssimati

- lezione 8 -

Codifica dell'informazione non numerica

Posso associare a una sequenza di bit se ho numero finito di oggetti. (immagini, colori ...)

CARATTERI

↳ sequenza di simboli, ne abbiamo qtà ≠ in base alla lingua

→ è un insieme finito di N elementi ⟹ avremo bisogno di $\log_2 N$ bit.

Avremo assegnazione che deve essere condivisa da tutti i calcolatori

↳ problema già posto con il Telegrafo in passato

⇒ codifica ASCII: ogni simbolo rappresentato da 1 byte (8bit)

(anche l'andare a capo è codificato)

↳ codificato da 8 bit ogni carattere

con 8 bit rappresento fino a 256 simboli (2^8)

⇒ UNICODE = estensione ASCII

organizzata su 2 byte → 2^{16} caratteri

file di testo in ASCII sono file.TXT = file rappresentato dalla sequenza ASCII dei caratteri

↓

Non abbiamo il font. È diverso da un file word che ci dice anche come è il carattere che scrivo.

Ad es. la lettera A ha un certo codice ASCII. Per determinare la forma, la grandezza ecc. c'è prima della sua codifica una sequenza di byte che determinano lo "stile".

(2 byte per come è fatto e un byte per codificare la lettera in sè ad esempio)

Esercizio: leggere numero da Tastiera e mandare messaggio se numero è pari o dispari.

```
main ()
{  int A, R ;
printf ("immetti un numero ") ;
scanf ("%d\n",&A) ;
R = A%2 ;
if (R== 0)
  { printf ("il numero è pari");}
else
  { printf ("il numero è dispari");}
}
```

Esercizio: leggere da Tastiera 3 numeri e valutare se possono essere lati di un Triangolo rettangolo.

```
main ()
{ int a,b,c ;
  printf ("numero = ") ;
  scanf ("%d", &a);
  printf ("numero = ") ;
  scanf ("%d", &b);
  printf ("numero = ") ;
  scanf ("%d", &c);
if (a*a ==b*b + c*c)
  { printf ("rettangolo"); }
else
  { if (b*b == a*a + c*c)
    { printf ("rettangolo"); }
   else
    { if (c*c == a*a + b*b)
      { printf ("rettangolo"); }
     else
      { printf ("no rettangolo"); }
    }
}
return 0;
```

Istruzione switch

= al fronte di una condizione ho tante possibilità.

Sintassi:

```
switch (espressione)        (n)
{
  case 1 :                  n = 1  indica il valore
    { blocco 1 }
    break ;
  case 2 :                  n = 2
    { blocco 2 }
    break;
  ...
  default:                  Se non si è verificata alcuna condizione scatta esegui il blocco default
    { blocco default } ;
```

{ } = alt gr + maiusc + è

Esercizio 2 Programma che dica se la variabile è positiva o negativa e se multipla di (N) (valore costante).

Per la seconda parte avevo bisogno di una costante che inserisco con #define

```
#define N 3
main()
{ int num = 123;
  if (num > 0)
  { printf ("valore positivo");
  }
  else { printf ("valore negativo");
  }
if (num % N == 0)
{ printf ("\n numero multiplo di N");
}
else {
printf ("numero non è multiplo di N");
}
```

Inizializzazione che non è visualizzazione da tastiera

\n in modo da andare a capo dopo il messaggio precedente

- Per scegliere il progetto che voglio utilizzare vado sul nome nella colonna sx, clicco dx e seleziono "attivate project".
- "workspace" può essere nominato e salvato come cartella. → Aprendolo poi trovo TUTTI i progetti del workspace.

Esercizio 3 Individuare il massimo tra 3 valori interi

```
int main()
{   int v1 = 1, v2 = 2, v3 = 3;
    if (v1 > v2) {
      if (v1 > v3) {
          printf ("valore max è v1");
          }
          else{("valore max è v3");
          }
      }
  else {
      if (v2 > v3) {
      printf ("il valore max è v2")
      }
    else { printf ("il valore max è v3")
      }
  }
```

Enunciazione → posso dire se è vero o falso.

↳ le correlo con le congiunzioni e creo frasi complesse. → le frasi composte come sono correlate con le frasi semplici che la compongono?

frasi f_1, f_2 correlate da "e", "o"

↳ "e", "o" sono operatori booleani (AND, OR)

= basi per la costruzione delle frasi

NON = operatore di inversione che passa da V a F o viceversa. = NOT

⇒ Voglio poi capire se la composta è V o F in base al valore delle singole frasi.

es. F and V → F (presume che entrambe le variabili f_1 e f_2 siano V.)

possibili combinazioni sono 4 →

AND

f_1	f_2	ff
F	F	F
V	F	F
F	V	F
V	V	V

In modo analogo posso creare altre due tabelle per OR e NOT

NOT

f_1	ff
V	F
F	V

OR

f_1	f_2	ff
F	F	F
V	F	V
F	V	V
V	V	V

Da qui posso definire un'algebra booleana (con lo stesso criterio dell'algebra classica, ma con solo 2 VARIABILI)

Operatori sono solo AND, OR, NOT

0, 1 (V, F)

$f = x$ AND y OR z è già funzione booleana

precedenza { NOT 1, AND 2, OR 3

{ x = F, y = V, z = F } → f? → F AND V OR F → F OR F → F

l'algoritmo è la Tabella che mi dà le combinazioni.

simboli

AND	* x
OR	+
NOT	$\bar{x}$, x'

→ $f = x * y + z$

{ AND = moltiplicazione logica, OR = somma logica

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 35 di 134

Altri **Teoremi base**

$A \times A' = 0$ (falso)

$A + A' \times B = A + B$

$A + A \times B = A$

$A + A' = 1$ (vero)

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

$A \times (A' + B) = A \times B$

posso giungere a risolvere delle **espressioni**

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

$x + \bar{x}y = x + y$ (Teorema)

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

$y + \bar{y} = 1$ (Teorema)

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

$1 +$ qualsiasi cosa $= 1$

$= (1)$

1) posso semplificare ricordando i teoremi

2) rappresento attraverso Tavola della verità

2) $x + \bar{y} + \bar{x}y + (x + \bar{y})\bar{x}y$ 2 variabili

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

$\Rightarrow f = 1$

prendo valori e sostituisco con righe della Tabella

→ $0 + \bar{0} + \bar{0} \cdot 0 + (0 + \bar{0})\bar{0}0 \ldots = (1)$

→ $0 + \bar{1} + \bar{0}1 + (0 + \bar{1})\bar{0}1$

$0 + 0 + 11 + (0 + 0)11$

$0 + 1 + 0 \cdot 1 \cdot 1 = 1 + 0 = (1)$

→ stessa cosa con ultime 2 righe

Esempio: capire se due espressioni sono uguali o diverse

$f_1 = f_2 \Rightarrow$ uguali Tabelle di verità

$(f_1) \quad ? \quad (f_2)$

$x\bar{y} + \bar{x} \stackrel{?}{=} x + y$

1) colonne = variabili
2) righe = combinazioni
3) sostituzione delle combinazioni nelle formule.

Costruzione Tabelle verità

x	y	f_1
0	0	1
0	1	1
1	0	1
1	1	0

- $0 \cdot \bar{0} + \bar{0}$ → $0 \cdot 1 + 1$ → $0 + 1 = 1$
- $0\bar{1} + \bar{0}$ → $00 + 1 = 1$
- $1\bar{0} + \bar{1}$ → $11 + 0 = 1$

x	y	f_2
0	0	0
0	1	:
1	0	:
1	1	

potevo aggiungere colonna all'altra Tabella

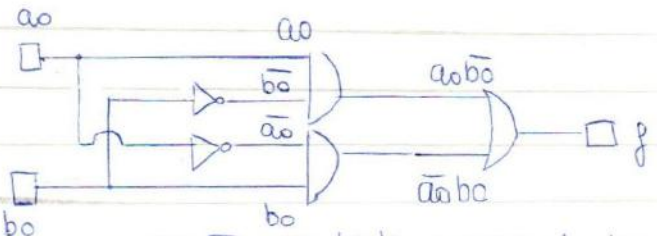
già il primo è diverso

$\Rightarrow$ diverse

$f = m_1 + m_2$

$= a_0 b_0 + a_0 b_0$

$= a_0 \bar{b_0} + \bar{a_0} b_0$



= circuito che fa la somma di due bit

Con lo stesso principio dei chip si creano le memorie. → Algebra booleana alla base della creazione di circuiti.

→ nel linguaggio C → operatori booleani si utilizzano per esplicitare le condizioni

AND = &&
OR = ||
NOT = !

Esempio:
(a > 5) && (a >= b)

esempio: programma può stampare come risultato VERO o FALSO

```
int a,b,c;
a = 5;
b = 3;
c = (a < b);
printf ("%d", c);
```

} verifica la condizione a<b e stampa 1 o 0 se è vero o falso

esercizio: leggere 4 valori che sono lati Trapezio, calcolare se è isoscele.
(Devo cercare almeno due numeri uguali)

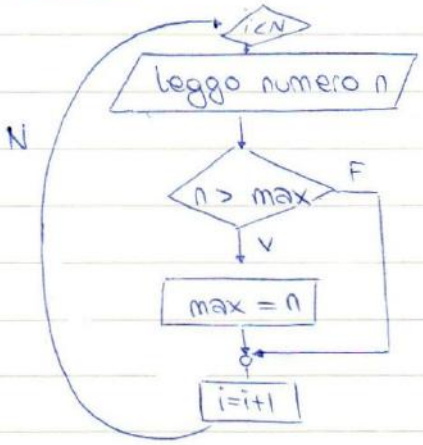
```
main ()
{ int a,b,c,d;
  printf ("lato = ");
  scanf ("%d", a);
  ... (x 4 volte quanti sono i lati)
```

Metto già TUTTE le 6 condizioni possibili

```
if ((a==b) || (a==c) || (a==d) || (b==c) || (b==d) || (c==d))
    printf ("isoscele");
else
    printf ("NO isoscele");
```

costante che definiamo

Esempio: calcolare valore max di N numeri interi da tastiera. Numeri interi positivi.



devo inizializzare i = 0
max = 0

il minor numero positivo all'inizio

```
#define N 4
main()
{ int i, max, n;
  i = 0;
  max = 0;
  while (i<N) {
    printf("numero=");
    scanf("%d", &n);
    if (n>max) {
      max = n;
    }
    i++;
  }
printf("max = %d", max);
return 0;
```

i++; → Dentro il while!

Se avesse chiesto numeri interi e non numeri interi positivi sarebbe stato sbagliato → NO max=0

Avrei dovuto mettere

```
max = INT_MIN
#include <limits.h>
```

- lezione 12 -

Esercizio 1 leggere da tastiera un numero di coppie di numeri (n° di coppie introdotto da tastiera) e calcolare la coppia che dà la somma max e la coppia che ha intervallo massimo.

Esercizio 2 $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} \ldots$ Calcolare e^x con N = 5 iterazioni

```
#include <limits.h>
int main()
{
   int i,n,c1,c2,max,maxi,inter,posm,posi;
      printf("n. coppie = ");
      scanf("%d",&n);
      i=0;
      max = INT_MIN;
      maxi = INT_MIN;
      while (i<n) {
            printf("valore 1 =");
            scanf("%d",&c1);
            printf("valore 2 =");
            scanf("%d",&c2);
            if ((c1+c2)>max)
             {max = c1+c2;
              posm = i;}
            inter = c1-c2;
            if (inter<0)
              inter = -inter;
            if (inter > maxi)
            {maxi = inter;
              posi = i;}
            i++;
           }
      printf("max=%d pos=%d -- maxi=%d pos=%d",
                  max,posm,maxi,posi);
```

max intervallo

posizioni

Inizializzazione

Inserimento dei numeri della coppia

posizione della coppia

cambio segno

Esercizio 3 Leggere delle Temperature da Tastiera e calcolare la Tmax, la Tmin e la Tmedia. Capire quanti valori sono Tra N1 e N2. (introdotti da Tastiera)

2 versioni (1) n° di Temperature costante → #define N

(2) " " " non definito ma si blocca quando introduco T = -1000°C

Esercizio 4 (a) 2 numeri su 8 bit in complemento a 2 scritti in esadecimale

F3 ; 7 1 → Valore decimale? Somma? overflow si/no?

(b) $f_1 = x\bar{y}\bar{z} + z\bar{y}$ → Tabella verità

$f_2 = \bar{x}(yz + \bar{y})$ $f_1 \stackrel{?}{=} f_2$

(a) Traduci prima in sequenza binaria

↳ ogni cifra su 4 bit

F 3 → 1111 0011 → 11110011 → Negativo → $-2^7 + 2^6 + 2^5 + 2^4 + 2 + 1 = -13$

7 1 → 0111 0001 → 01110001 → Positivo → $2^6 + 2^5 + 2^4 + 1 = 113$

oppure posso calcolare con la complementazione

11110011 → 00001100 + 00000001 = 00001101 = 13 → quindi numero di partenza che era il suo opposto era +13

↓

Inutile utilizzare secondo metodo. (già positivo)

Overflow? ⟹ Impossibile poiché hanno segno opposto, la somma dà modulo minore

Somma = +100 → Devo scrivere in binario

11110011 + 01110001 = 01100100

oppure $100 = 64 + 32 + 4 = 2^6 + 2^5 + 2^2$

01100100

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 45 di 134

```
#include <limits.h>
int main ()
{
   int i,t,max,min,media,nint,n1,n2;
   float media;
   do{ printf ("n1 = ");
       scanf ("%d", &n1);
       printf ("n2 = ");
       scanf ("%d", &n2); }
   while (n1 >= n2)
       i = 0;
       media = 0;
       nint = 0;
       max = INT_MIN;
       min = INT_MAX;
       printf ("Temp = ");
       scanf ("%d", &t);
   while (t != -1000){
       if (t > max) {
          max = t;
         }
       if (t < min) {
          min = t;
         }
       if ((t >= n1) && (t <= n2))
          nint++;
       media = media + t;
       i++;
       printf ("Temp = ");
       scanf ("%d", &t);
       }
   media = media / i;
   printf ("%d ; %d ; %d ; %f ", max, min, nint, media);
```

numero di Temperature che sono nell'intervallo n1÷n2

no intero

condizione che non mi va bene per la quale torno all'inizio

inizializzazione

inserimento Temperatura

Condizioni di massimo e minimo

condizione perché sta nell'intervallo

nuovo inserimento Temperatura

sarebbe da fare se e solo se i ≠ 0 → Altrimenti non fa la media!

Utilizzare istruzione do

Scrivo prima il blocco e poi while (n1 >= n2), devo rileggere n1 e n2 se non vanno bene.

Sintassi

```
do
  { blocco }
while (condizione);
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 47 di 134

```
printf ("\n")          → andare a capo all'altine del lato
i = 0;
while (i < N-2) {
    printf ("*");
    j = 0;
    while (j < M-2) {
        printf (" ");
        j++;
        }
    printf ("*\n");
}
i = 0;
while (i < M) {
    printf ("*");
    i++;
}
printf ("\n");
return 0;
```

Ricordarsi TUTTE le inizializzazioni

Lati verticali con while annidati per la creazione degli spazi

Riga di coda

ISTRUZIONE FOR — lezione 14

= Forma compatta per scrivere il while

```
i = 0      Inizializz.
while (condizione)
{ .... }
i++        fine ciclo
```

→

```
for (inizializz; condiz.; fine ciclo)
{ ... }
```

Esempio — leggere numero binario (N bit) e calcoliamo decimale

so che $N = \sum b_i \cdot 2^i = b_0 \cdot 2^0 + b_1 \cdot 2^1 + \dots$ per calcolare

Indice di ciclo i = esponente di 2

→ somma $s = s + b * 2^i$

```
# define N 5
{ int a0, a1, a2 ;
    a0 = 0 ;
    a1 = 1 ;
    for ( i=0 ; i<N ; i++ ) {
        printf ("%.d\n", a0 ) ;
        a2 = a1 + a0 ;
        a0 = a1 ;
        a1 = a2 ;
    }
    return 0 ;
}
```

Inizializzare forzando i primi due risultati, 0 e 1.

← Ad ogni giro stampa il nuovo a0

⟹ devono cambiare valori per calcolo successivo

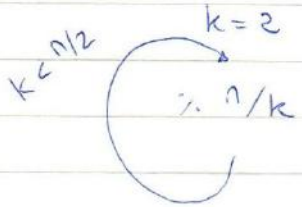
Esercizio Introduco numero (n), calcolare se n è primo

prendere n → dividerlo per ciò che c'è prima

numeri k che vanno da 2 a n/2.

Ripetere n % k finché non Trovi resto = 0.

Programma si deve fermare se arrivi a n/2 o se resto = 0.



Devo avere una variabile per quanto riguarda il resto.

flag che mi fa passare al k successivo se resto ≠ 0. Che mi fermi se resto = 0

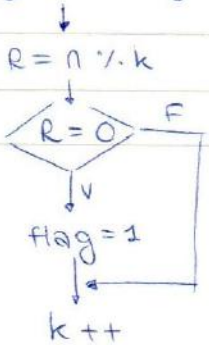
flag = 0 ⇒ Resto ≠ 0

flag = 1 ⇒ Resto = 0

↓

flag è "segnalino"

Ripeto il ciclo se $k < \frac{n}{2}$ && flag = 0



leggere ci
" r
" a

applicare direttamente la formula
$n = 1 + r/100$
$c_f = c_i * pow(n,a)$
n deve essere float
→ mettere (float) prima di r/100

non applicare la formula della potenza → utilizzare for

for (i=0; i<a; i++)
$(1+r)^k = (1+r)(1+r)^{k-1}$
sarà
$p = 1$
$p = p * (1+r)$

Nota che la variabile di ciclo in questo caso non è stata utilizzata nella formula → conta solo i giri.

```
int ci, r, a;
float p, cf;
printf ("capitale iniziale=");
scanf ("%d", ci);
printf ("Tasso annuo =");
scanf ("%d", r);
printf ("anni =");
scanf ("%d", a);
```

Inserire do { } while in modo che con l'inserimento di 0 richieda di nuovo di mettere gli anni

```
p = 1;
for (i = 0; i < a; i++) {
    p = p * (1 + (float) r/100);   // UTILIZZARE
    cf = ci * p;                   // NON cambiare cf
    printf ("anno = %d  cf = %f\n", i, cf);
}
```

Stampa una riga per ogni anno in tal modo.

per evitare che stampi "anno 0" metto i+1.

Esercizio Le calcolatrici come calcolano la radice quadrata?

Sviluppo in serie iterativo

$$x_n = \left(x_{n-1} + \frac{N}{x_{n-1}}\right)/2$$ che converge a $\sqrt{N}$ con $n \to \infty$

(1) quando mi fermo? all'infinito $|x_n - x_{n-1}| = 0$
↳ per non andare all'infinito accettiamo errore ragionevole $\varepsilon = |x_n - x_{n-1}| = 0{,}01$ che sarà l'approssimazione. Programma che si ferma quando l'errore è <= di quello

(2) x_0? Da dove parto?
→ qualcosa < N
N/2, N/3...

$$\begin{cases} x_2 = \left(x_1 + \frac{N}{x_2}\right)/2 \\ d = x_2 - x_1 \end{cases}$$

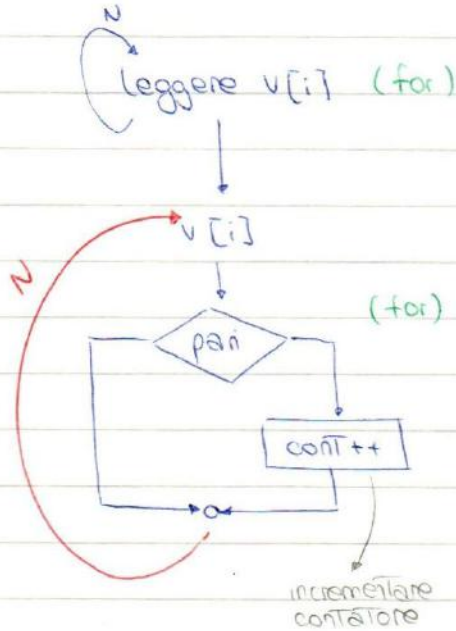
Da tastiera leggere N, ε.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 53 di 134

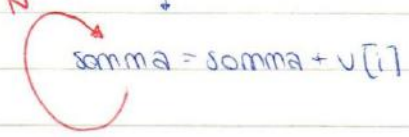
Possiamo utilizzare la singola scatolina come variabile.

Esempio leggere da Tastiera un vettore di 3 elementi e calcolare:

- quanti sono pari
- sommatoria dei valori



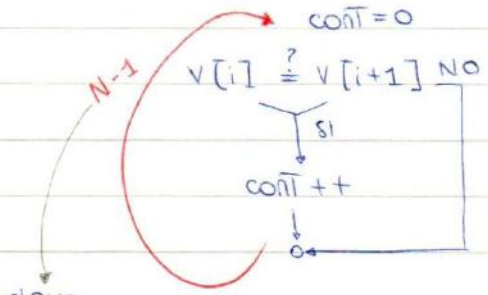

```
#define N 4
{ int v[N], i, cont, somma          // cont → numero di numeri pari
  for(i=0; i<N; i++) {
    printf("v[ ] = ");              // lettura del vettore
    scanf("%d", &v[i]);
  }
  cont = 0;
  for (i=0; i<N; i++){
    if (v[i]%2 == 0)
      cont++;
    }
printf("cont = %d\n", cont);
```




```
  somma = 0;
  for (i=0; i<N; i++){
    somma = somma + v[i];
    }
printf("somma = %d", somma);
return 0;
}
```

Esercizio Vettore di N elementi interi. Coppie di elementi adiacenti sono uguali? quanti? - lezione 18 -

Devo guardare coppie di elementi adiacenti. (v[i] e v[i+1])
Verificare se essi sono uguali o meno → Aggiornare contatore.

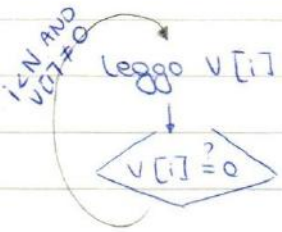


Valori predefiniti che do al vettore: inizializzazione vettore

```
#define N 8
main()
{ int i, cont, v[N] = {2,3,3,5,4,4,4,8};
  cont = 0;
  for (i=0; i<N-1; i++){
    if (v[i] == v[i+1])
      cont++;
  }
printf ("cont = %d", cont);
return 0;
}
```

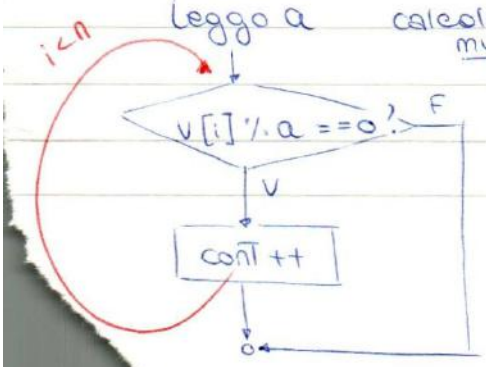
dentro if posso mettere il printf per scrivere quali sono i numeri

Esercizio Leggere da tastiera un vettore di al max N elementi. Termine del vettore quando arrivo a N elementi o quando immetto valore = 0. Calcolare quanti sono multipli di un valore dato da tastiera.



Usiamo do-while

Non perfetto perché dobbiamo tenere conto degli elementi che risultano validi, se introduco 0 esso non vale. Devo incrementare i++ solo quando ho introdotto v[i] ≠ 0.




```
#define N 5
main()
{ int i, v[N], cont, a, n;
  i=0;
  do{
    printf ("v[] = ");
    scanf ("%d", &v[i]);
    if (v[i] != 0)
      i++;
  }
while((i<N) && (v[i] != 0));
n = i;
printf ("valore = ");
scanf ("%d", a);
cont = 0;
for (i=0; i<n; i++){
  if (v[i] % a == 0)
    cont++; }
printf ("cont = %d", cont);
```

Introduzione dei dati

Solo Termini validi (=N se non introduco mai 0 ∅)

```
#define N 12
main()
{
    int i,j,k
    for (i=0; i<N; i++) {
        for (j=0; j<i; j++) {
            printf (" ");
            }
        for (k=0; k<N-i; k++) {
            printf ("*");
            }
    printf ("\n");
    }
 return 0;
}
```

Abbiamo 3 for dunque 3 variabili

devo fare solo i spazi

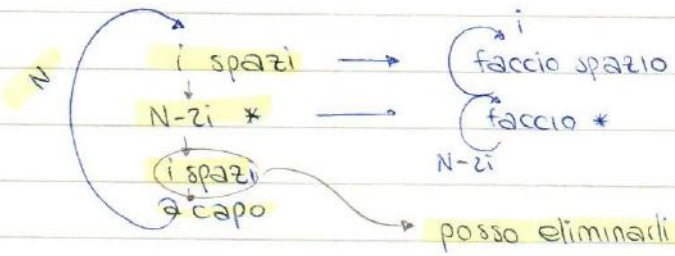
Creazione della riga

Avrei potuto utilizzare nuovamente j poiché non venivano usate in contemporanea.

(b) Abbiamo N righe

→ seconda riga = spazio + (N-2) * + spazio

Terza riga = 2 spazi + (N-4) * + 2 spazi



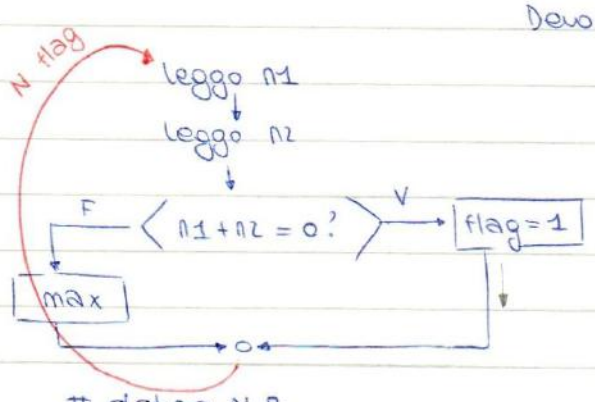
posso eliminarli poiché a destra è già vuoto

```
for (i=0; i<N; i++) {
    for (j=0; j<i; j++) {
        printf (" ");
        }
    for (j=0; j<N-2*i; j++) {
        printf ("*")
        }
```

Non ho utilizzato k come detto prima

Cambia solo questa istruzione

Esempio leggere coppie di numeri da Tastiera (n1, n2). Numeri interi. Calcolare coppia il cui $(n1^2 + n2^2) = max$. Vogliamo fermarci dopo N volte oppure quando $n1 + n2 = 0$ ⟹ for condizionato



Devo ripetere in funzione di due condizioni:

① $i < N$ ② $flag = 0$

condizione = (i < N && flag == 0)

posizione della coppia con massimo

```
# define N 3
# include <limits.h>
main()
{ int i, n1, n2, flag, max, imax
  flag = 0;
  max = INT_MIN;
  for (i = 0; i<N && flag==0; i++) {
    printf ("n1 = ");
    scanf ("%d", &n1);
    printf ("n2 = ");
    scanf ("%d", &n2);
    if (n1 + n2 == 0) {
      flag = 1;
    }
    else {
      s = n1 * n1 + n2 * n2;
      if (s > max) {
        max = s;
        imax = i+1;
      }
    }
  }
```

printf ("\n"); per spazio Tra coppie

→ Non abbiamo tenuto conto delle condizioni al contorno per cui se esco già con la prima coppia non mi dà risultato del max.

↳ if (flag == 1)
max = 0 ---> printf scanf

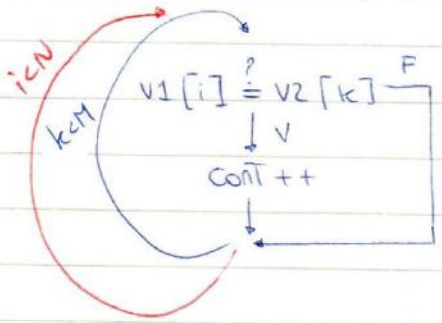
in modo che la prima coppia sia 1 e non 0

in modo da vedere per cosa siamo usciti.

```
" max = %d   imax = %d"   i = %d", max, imax, flag);
```

Esercizio 2 vettori V1[N] e V2[M]. V1 definito, V2 letto da tastiera. Quanti elementi di V1 sono anche in V2?

Devo prendere ogni elemento di V1 e confrontarlo con TUTTI quelli di V2.




```
#define N = 6
#define M = 3
int main()
{ int i, k, V1[N] = {1,2,3,4,5,6}, v2[M], cont
    for (i=0; i<M; i++) {
    printf("v2[] = ");
    scanf("%d", v2[i]); }
    cont = 0;
    for (i=0; i<N; i++) {
    for (k=0; k<M; k++) {
      if (v1[i] = v2[k])
          cont++;
      }
    }
  printf("cont = %d", cont);
return 0;
}
```

Inserimento v2

Se vettore v2 non è lungo M ma può avere M elementi oppure averne meno se digito $. Come cambia programma?

do-while

Importante: compito

ORDINAMENTO di un vettore

17	3	5	12	7
17	12	7	5	3

Il primo lo trovo calcolando il max e mettendolo in prima posizione. Poi prendiamo solo gli elementi rimasti, calcolo max e lo sostituisco nella seconda posizione e così via. Considero via via un vettore sempre più piccolo.

i = 0

Calcolo il max a partire da i → max = v[primo]

↳ Iterativo sul vettore

Sposto il max nella prima posizione(i)

i < N-1

l'ultima verifica è quella dell'ultima coppia.

```
#define N 5
main()
{ int i,k,posm,max,Temp, v[N]={4,3,7,2,1}
  for (i=0; i<N-1; i++) {          → fino a ultima coppia
    max = v[i];                    ← Ad ogni giro imposto il primo elemento che analizzo come il massimo
    for (k=i; k<N; k++) {          → parte da i e non da 0
      if (v[k] > max) {            ← Calcolo del max
        max = v[k];
        posm = k;                  → posizione del max
        }
    }
    Temp = v[i];
    v[i] = v[posm];                ← Mettere il max nella giusta posizione
    v[posm] = Temp;                → Scambio di posizioni con il massimo: metto max al primo posto disponibile
}
for (k=0; k<N; k++)
  printf("v[]=%d\n", v[k]);
```

Ho preso v1 e l'ho Tiascritto in v3.

Per caricare v2 devo partire da np+1 e andare avanti per non coprire quelli già scritti.

```
for (i=0; i<np; i++)
    v3[i] = v1[i];
for (i=0; i<nd; i++)
    v3[np+i] = v2[i];
printf("\n");
for (i=0; i<nd+np; i++)
printf("v3[] = %d\n", v3[i]);
```

lunghezza vettore v3

Scrittura del vettore v3

Dobbiamo ora ordinare v3 con il principio del massimo (se vogliamo decrescente) e minimo (se vogliamo crescente).

```
// ordinamento
for (i=0; i<np+nd-1; i++) {
  max = v3[i];
  for (j=i; j<np+nd; j++){
    if (v3[j] >= max){
      max = v3[j];
      posm = j;
    }
  }
  Temp = v3[i];
  v3[i] = v3[posm];
  v3[posm] = Temp;
}
```

Registra massimo e posizione

In generale

```
for (i=0; i<N-1; i++)
{ Trovo max
  scambio max }
```

max = primo elemento = v[i]
for (j = primo elemento ...)

Scambio posizione

3	7	9	1	5	4

prendiamo primo giro i=0

Alla fine di quel for avrò che max = 9 e posm = 2

Temp = v3[0] Temp → 3
v3[0] = v3[posm] v3[0] → 9
v3[posm] = Temp v3[posm] → 3

Così ho scambiato i valori

```
printf("\n");
for (i=0; i<np+nd; i++)
  printf("v3[] = %d\n", v3[i]);
```

Al secondo giro sarà v3[i] = v3[1]

quindi non Tocco più il massimo che ho già Trovato.

```
#define N 4
int main()
{ int h[N], i, k, posm
  float t[N], max, min, Temp
```

variabile per lo scambio di posizione

Temp usato sia come float che int

```
for (i=0; i<N; i++) {
  printf("Temperatura = ");
  scanf("%f", &t[i]);
  printf("ora = ");
  scanf("%d", &h[i]);
  }
```

Inserimento di TUTTI i valori

```
max = t[0];
```

max al primo elemento

```
for (i=0; i<N; i++) {
  if (t[i] >= max)
    max = t[i]; }
min = t[0];
```

min al primo elemento

```
for (i=0, i<N; i++) {
  if (t[i] <= min)
    min = t[i];
  }
printf("\n");
printf("escursione = %f", max-min);
```

\n\n

meglio N-1

```
for (i=0; i<N; i++) {
  min = h[i];
  for (k=1; k<N; k++) {
    if (h[k] <= min) {
      min = h[k];
      posm = k;
    }
  }
```

Ordinamento del vettore in base all'orario.

chiuso questo for sappiamo dove sta min

```
Temp = h[i];
h[i] = h[posm];
h[posm] = Temp;
```

Scambio le Temperature seguendo le ore!

```
Temp = t[i];
t[i] = t[posm];
t[posm] = Temp;
}
for (i=0; i<N; i++)
  printf("ora = %d  t = %f\n", h[i], t[i]);

  return 0;
}
```

Le funzioni

Funzioni abituali sono composte da milioni di righe di codice → l'unica soluzione è comporre in blocchi le funzioni. → sottoprogrammi che svolgono una "semplice" funzione. In più abbiamo funzioni già registrate in libreria → Moduli riutilizzabili.

Le soluzioni MONOLITICHE di miei grandi programmi non sono gestibili

↳ Approccio "TOP-DOWN" = scomposizione di un problema in sotto-problemi + semplici

Vale anche per la scrittura dei programmi

↳ MAIN con una serie di moduli che svolgono funzioni specifiche.

Moduli o sottoprogrammi chiamati:
- funzioni
- procedure
- subroutine

Moduli vengono richiamati dal main e utilizzati.

Una funzione riceve dei valori in ingresso e ne fa uscire di nuovi

parametri ⇒ f ⇒ risultato

(ad esempio sqrt (a) fa radice quadrata)

Chiamate di funzioni

- La funzione deve essere denominata e viene richiamato dal suo nome

(1) Dichiarazione ("Io voglio adoperare questo") prima del main (o prototipo)

```
int cubo (int a);
```

(2) Nel corpo del main richiamiamo 1 o + volte la funzione.

anche float o void (che non fa tornare risultato)

```
int cubo ( int a )
```

- di che tipo è il risultato che mi dà la funzione
- nome
- Tipo e nome della variabile che uso dentro il corpo della funzione.

"scatola che ricevo alla fine" — "scatola che devo utilizzare"

(3) Alla fine del main scrivo la DEFINIZIONE della funzione ovvero come è fatta. (o corpo)

```
int cubo (int a) {
int c;
    c = a * a * a;
    return c;
}
```

Non la dichiaro nel main perché è una scatola che utilizzo solo qui.

Istruzione per far passare la scatola indietro il "return" manda indietro valore contenuto o risultato.

Esercizio Creare funzione uguale a pow (b,a): funzione "expo".

(1) dichiarazione

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

devo passare due valori poiché devo utilizzare base ed esponente

(3) Composizione del main

```
int main()
{ int x,y, risultato;

printf("base=");
scanf("%d",x);
printf("esponente=");
scanf("%d",y);
```

(4) chiamata f(x)

```
risultato = expo(x,y);
printf("risultato=%d", risultato);
return 0;
}
```

In alternativa

```
printf("risultato=%d", expo(x,y));
```

saltando una variabile

(2) corpo

```
int expo (int a, int b) {
int i, prod;
prod = 1;
for (i=0; i<b; i++){
    prod = prod * a;
    }

return prod;
}
```

Esercizio funzione f che riceve coppia di numeri e vede se sono primi tra loro

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

Risultato = 0 non sono primi

= 1 sono primi

Esercizio Piano cartesiano con due segmenti $\overline{P1P2}$ e $\overline{P3P4}$. -lezione 23-

Introdurre le coordinate da tastiera e calcolare con una funzione quale segmento è più lungo.

Leggere le coordinate
↓
chiamare funzione
ℓ_1 = lato (P1, P2)
ℓ_2 = lato (P3, P4)
↓
Calcolo max

funzione lato?

distanza tra due punti con coordinate
x1, y1 ; x2, y2.

Dichiarare funzione
float lato (int x1, int y1, int x2, int y2)
↳ poiché dobbiamo fare radice

```
#include <math.h>
float lato (int x1, int y1, int x2, int y2);
int main()
{ int p1x, p1y, p2x, p2y, p3x, p3y, p4x, p4y;
  float e1, e2;
  printf ("p1x =");
  scanf ("%d", &p1x);
  printf ("p1y =");
  scanf ("%d", &p1y);
```
} Ripetuto 4 volte per i 4 punti

```
  e1 = lato (p1x, p1y, p2x, p2y);
  e2 = lato (p3x, p3y, p4x, p4y);
```
} Utilizzo due volte la funzione ne scrivendola solo 1 volta

```
  if (e1 > e2)
    printf ("max P1P2");
  else
    printf ("max P3P4");
  return 0;
}
```

```
float lato (int x1, int y1, int x2, int y2) {
float l;
l = sqrt ((y2-y1)*(y2-y1) + (x2-x1)*(x2-x1));
return l;
}
```

punta a scatola con valore reale

Devo passare anche il puntatore

`float ff (int a, int b, float *p)`

↳ In base a che tipo di risultato desideravo dalla divisione

Nel main avrò bisogno di due variabili che tengano conto dei due risultati. ("q" e "perc").

```
main ()
{
int x,y
float q, perc
------- x
------- y
q = ff (x, y, &perc)
```

(b) fornisco direttam. l'indirizzo

(a) esplicitato puntatore

oppure avrei potuto

```
float *pp;
....
pp = &perc;
q = ff (x,y, pp);
```

↳ scrittura del programma

```
float ff (int a, int b, float *p);
main ()
{ int x,y;
  float q, perc;
  x = 3;
  y = 7;
  q = ff (x,y, &perc)
  printf ("quoziente = %f   perc = %f \n", q, perc);
  return 0;
}
```

perc = variabile
&perc = indirizzo variabile
*p = puntatore che va a perc

No lettura da tastiera

chiamata della funzione

```
float ff (int a, int b, float *p) {
float div;
div = (float) a/b;
*p = a*b*0,1;
return div;
}
```

divisione

Scrivo già l'indirizzo della scatola dove c'è la variabile che dovrò riprendere nel main. (perc)

&perc nel main

q nel main

Esercizio 1 Dato un vettore, scrivere funzione che ritorna il max, il min e la varianta. (scarto quadratico medio).

Esercizio 2 Radar che fa una riga di N elementi. Ogni elemento è un punto dello spazio che vale 0 se non c'è ostacolo, vale 1 se c'è.
(a) Numero di ostacoli che trova (ogni serie di 1)
(b) ostacolo più grande (quello con più 1)

Esercizio 1

float ff (int v[], int N, int *pmin, int *pmax);

int main ()
{ int a[N] = {1,2,3,4,5}, max, min;
  float va;
  va = ff (a, N, &min, &max);
  printf ("varianza = %f \n", va);
  printf ("max = %d \n", max);
  printf ("min = %d", min);
  return 0;
}

float ff (int v[], int N, int *pmin, int *pmax) {
int i, sum;
float tt, mm, vv;
sum = 0;
for (i=0; i<N; i++)
    sum = sum + v[i];
mm = (float) sum/N;
tt = 0;
for (i=0; i<N; i++)
  tt = tt + (mm - v[i])(mm - v[i]);
vv = sqrt(tt)/N;
*pmax = v[0];
for (i=0; i<N; i++){
    if (v[i] >= *pmax)
        *pmax = v[i]; }
*pmin = v[0];
for (i=0; i<N; i++){
    if (v[i] <= *pmin)
        *pmin = v[i];
    }
return vv;
}

CARATTERI — lezione 25

Gestione dei caratteri → codice ASCII

Valore numerico di 1 byte che identifica caratteri e punteggiatura

Ad esempio y = 121 (decimale)

7 = 55 (decimale, 7 come carattere)

Insieme di caratteri = STRINGA di codici ascii corrispondenti ad ogni carattere presente

I/O di CARATTERI

In lettura: (1) scanf ("%c", &c) — introdotta come char c; ↳ scatola c di tipo char.

(2) getchar (c)

In stampa: (1) printf ("%c", c)

(2) putchar (c)

Esistono una serie di funzioni che permettono di manipolare caratteri

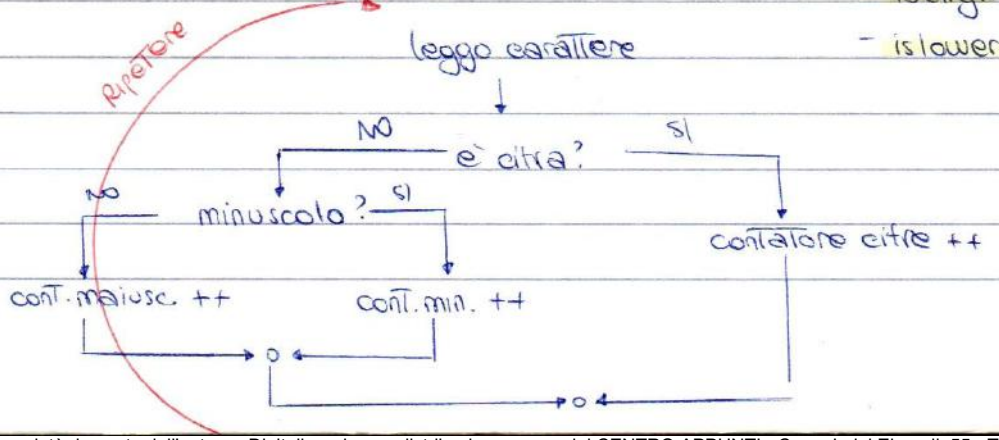
per utilizzarle <cType.h>

Funzioni più importanti:

int isalpha (char c)	→ dice se è lettera o meno
" isdigit (")	→ dice se è una cifra
" islower (")	→ " " minuscolo
" isupper (")	→ " " maiuscolo
" Tolower (")	→ forza a minuscolo
" Toupper (")	→ " " maiuscolo

Esempio: leggere 4 caratteri e vedere quanti sono cifre. Delle non cifre quanto sono minuscole.

funzioni:
- isdigit
- islower



dere ciascun elemento di c2
Pre confrontarlo con TUTTI elementi
e di c1 → doppio for

Torna già da solo se è
RI vero o falso. esegue l'if
se vera (non mettere != 0)

Rendo maiuscolo e minuscolo
lo caratteri di c1 in base a
come sono quelli di c2. Uso
variabile "c" per non modificare
il vettore c1.

```
#include <ctype.h>
#define N 5
#define M 3
int main()
{
 char c1[N] = {'m','e','z','z','A'}, c2[M], c;
 int i,k, cont;
 for(i=0; i<M; i++){
  printf("car =");
  scanf("%c%*c", &c2[i]);}
   for(i=0; i<M; i++){
    for(k=0; k<N; k++){
     if(islower(c2[i]))
      c = Tolower(c1[k]);
     else
      c = Toupper(c1[k]);
     if(c2[i] == c){
      printf("%c\n",c);
      cont++;}
    }
   }
 printf("cont = %d", cont);
 return 0;
}
```

Introdotti con apici

cont = 0;

prima c2 e poi c1

Lo rendo maiuscolo o minuscolo in base a come è c2. così da poterli confrontare. Darebbe sempre ≠ altrimenti.

stampare carattere che trova uguale

-lezione 26-

Esercizio Vogliamo cifrare un Testo con algoritmo di Cesare. Basato su chiave crittografica che è un numero (n=2 ad es.) che deve rimanere segreto. Ogni lettera scritta è spostata da quella vera di tante posizioni quanto indica la chiave. Applicare algoritmo a stringa.

'a' 'c' = messaggio originale
n=2 ↓ ↓
'c' 'e' = messaggio modificato

s = messaggio da codificare
s1 = messaggio codificato
es. chiave (3) 'b' → 'e'
'y' → vado oltre la z e devo riuscire a tornare all'inizio

Esercizio Chiediamo all'utente di inserire una sequenza. Quando incontriamo un punto smettiamo di leggere.

Potevo non utilizzare vettore e utilizzare caratteri lavorando sul singolo carattere alla volta e non sulla sequenza*

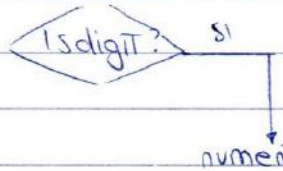
```
#include <ctype.h>
#define N 100   spazio massimo molto grande

int main()
{ char v[N];
  int i=0;
  int lunghezza, numeri
  printf("sequenza di caratteri Terminata da . \n");
  do{
  v[i] = getchar();
  i++;
  } while((i<N) && (v[i-1] != '.'));
```

elemento i-1 poiché abbiamo incrementato i++ alla riga prima. L'elemento i è quello che sto per scrivere al passaggio successivo.

```
  lunghezza = i;
  printf("Ho letto:");   → Dichiaro cosa ha registrato il programma
  for (i=0; i<lunghezza; i++) {
      putchar(v[i]);   Introduce tutti i caratteri
  }
```

Conteggio delle cifre




```
  numeri = 0;
  for (i=0; i<lunghezza; i++) {
    if (isdigit(v[i]) != 0) {
      numeri++;
    }
  }
  printf("Ho Trovato %d cifre", numeri);
```

1 = vero
0 = falso

* programma sarebbe cambiato → poiché la cosa fondamentale è capire se è fondamentale conoscere la sequenza o poter trattare caratteri uno per uno

```
...
char v;
int numeri;

printf("sequenza di caratteri Terminata da . \n");
do{
    v = getchar();
    if (isdigit(v) != 0)
      numeri++;
    }
while (v != '.');
printf("Ho Trovato %d cifre", numeri);
```

esercizio Leggere stringa da tastiera. Stampare lunghezza, contare caratteri che sono delle cifre. Tradurre tutte le maiuscole in minuscole → Tolower

Esercizio con "scanf" o "gets"
- lunghezza → strlen
- scorrere → da s[0] a s[N] utilizzando "isdigit"
- maiuscole → prendo di nuovo il singolo elemento, verifico se è maiuscolo con "isupper" e lo tolto con "Tolower"

```
#include <ctype.h>
#include <string.h>
#define N 80
int main()
{ int e, i, ndig
  char s[N+1];
    printf("stringa=");
    scanf("%s", s);
    e = strlen(s);
    printf("%d\n", e);

    ndig = 0;
    for(i=0; i<e; i++){
      if(isdigit s[i] != 0)
        ndig++;
    }
    printf("%d\n", ndig);
    for(i=0; i<e; i++){
      if(isupper s[i] != 0)
        s[i] = Tolower(s[i]);
    }
  printf("%s", s);
  return 0;
}
```

(così tengo conto di "\0")
(conto le cifre)
(Rendo minuscole)

Esercizio leggere una stringa del tipo:

<nome>	␣	U/D	␣	<altezza>

in base al sesso

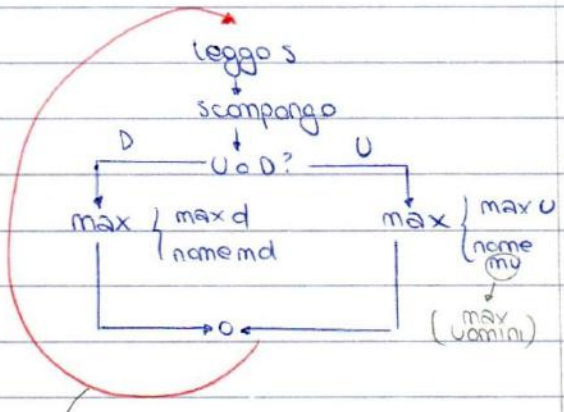
Vogliamo calcolare l'uomo e la donna più alti. Dovremo lavorare sul sesso e l'altezza.

Dovremo scorporare i 3 elementi della stringa (nome, sesso, altezza) → str, char, int

Suddividio in 3 parti coerenti con la tipologia che richiedono.

sscanf (stringa partenza, fattore di conversione, variabili)

Devo inserire anche i delimitatori (gli spazi nel nostro caso)
= "%s ␣ %c ␣ %d"



posso finire dopo N definito oppure variabile con un segno di stop ⇒ do-while

while è sempre condizione per cui torno da capo!

```
#include <string.h>
#define N 80                                   lunghezza stringa
{ char s[N], nome[N], gen, nomemu[N], nomemd[N];   sovradimensionato
  int maxu, maxd, a;                           altezza
  maxu = 0; maxd = 0;
  do {
    printf("stringa=");
    gets(s);                                   Se la stringa NON è "fine"
    if (strcmp(s,"fine") != 0) {
    sscanf(s,"%s %c %d", nome, &gen, &a);
    if (gen == 'u') {                          Non ha & poiché non è dato singolo
      if (a >= maxu) {                         UOMINI
        maxu = a;
        strcpy(nomemu, nome);                  Metto il nome del max in una nuova stringa
        }
      }
    else {
      if (a >= maxd) {                         DONNE
        maxd = a;
        strcpy(nomemd, nome);
        }
      }
    }
  }
while (strcmp(s,"fine") != 0);                 Eseguire comando fino a "fine"
printf("uomini = %s  altezza = %d\n", nomemu, maxu);
printf("donne = %s  altezza = %d\n", nomemd, maxd);
return 0;
}
```

```
for (i=0; i<R; i++) {
    parir = 0;
    for (j=0; j<C; j++) {
        if (m[i][j] % 2 == 0)
            parir ++;
    }
    printf ("n. elementi pari = %d in riga %d \n", parir, i+1);
}
```

per le righe

Dobbiamo scandire ora per colonne e poi per righe.

```
for (j=0; j<C; j++) {
    paric = 0;
    for (i=0; i<R; i++) {
        if (m[i][j] % 2 == 0)
            paric ++;
    }
    printf ("n. elementi pari = %d in colonna %d \n", paric, j+1);
}
return 0;
}
```

paric = 0; → deve essere dentro il for perché il conteggio deve essere azzerato all'inizio di ogni colonna

Stampa di una matrice

Immessi con # define

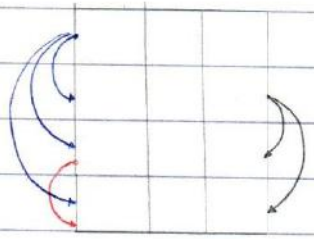
```
printf ("Matrice: %d x %d \n", N, M);
for (i=0; i<N; i++) {
    for (j=0; j<M; j++) {
        printf ("%f", mat[i][j])
    }
    printf ("\n");
}
```

Stampa della riga i-esima

Stampa riga x riga l'intera matrice.

Esercizio Matrice con R e C dimensioni. Scoprire se la matrice contiene due righe uguali tra loro.

⇒ È la ricerca di due vettori uguali all'interno della mat.
→ doppio ciclo for



i = riga di riferimento
j = righe che confronto con i

```
for (i=0; i<R-1; i++){
  for (j=i+1; j<R; j++){
```

Devo confrontare le righe all'interno
↳ ora usiamo flag
devo confrontare i con le successive
→ ciclo for per muovermi nelle colonne → k

```
  uguali=1;
  for (k=0; k<C; k++){
    if (m[i][k] != m[j][k])
      uguali = 0;
    }
```

flag=1 se uguali. Se mi viene =0 è perché sono diversi e posso terminare.

```
# define R4
# define C3
int main()
{ int m[R][C] = {{2,2,2},{1,2,3},{2,2,2},{3,4,5}};
  int i,j,k, uguali

  for (i=0; i<R-1; i++){
```

Parto a confrontare dalla riga successiva a quella considerata!

```
    for (j=i+1; j<R; j++){
      uguali = 1;
```

→ Inizializzo flag

```
      for (k=0; k<C; k++){
        if (m[i][k] != m[j][k])
          uguali = 0;
```

→ Riga i e i+1 in pratica

```
        }
      if (uguali == 1)
        printf("Ho almeno due righe uguali:
                %d e %d \n", i, j)
      }
  }

  return 0;
}
```

- Se voglio che trovando 3 righe uguali il programma si fermi alle prime due e lo comunichi?
- Se voglio cercare colonne uguali?

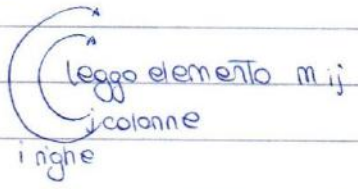
Cambiare ordine degli indici del ciclo for

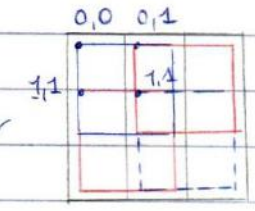
Cambiare le condizioni del ciclo for

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 93 di 134

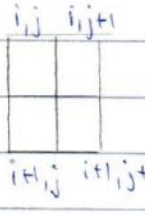
Esercizio Matrice RxC. Supponiamo che contenga interi. Voglio trovare di tutte le sottomatrici 2x2 quelle con sommatoria = 0. - lezione 30 -

leggere da tastiera i valori





Muovo l'elemento di testa
$i = 0 \longrightarrow i = R-1$ (for i)
$j = 0 \longrightarrow j = C-1$ (for j)
i e j saranno la partenza della sottomatrice. Devo considerare ora gli elementi della sottomat.



Posso scrivere tutti in modo esplicito (va bene per le 2x2, non se sono più grandi). ⇒ Altri due cicli for per muovermi nella sottomatrice.

```
#define R 3
#define C 3
int main ()
{ int m[R][C]
  int i,j, sum, k,h

  for (i=0; i<R; i++){
    for (j=0; j<C; j++){
      printf ("elemento m[%d][%d] = ", i,j);
      scanf ("%d", &m[i][j])        → C va perché è un singolo elemento.
    }
  }

  for (i=0; i<R-1; i++){
    for (j=0; j<C-1; j++){
      sum = 0;
      for (k=i; k<i+2; k++){
        for (h=j; h<j+2; h++){      → calcolo della somma
          sum = sum + m[k][h];
        }
      }
      if (sum == 0)
        printf ("somma = 0 in %d, %d\n", i, j);
    }
  }

  return 0;
}
```

[Se metto r2 → calcolo somma della seconda riga]

riga/colonna

Esercizio Leggere da tastiera un numero seguito da "r" o "c" e mediante la funzione voglio calcolare la somma degli elementi della riga o della colonna.

Come si passa funzione a matrice?

Dichiarazione

colonne esplicitate

```
int ff (int m[][C], int RR, char k, int q)
```

passiamo carattere e numero "r" o "c" + numero

passiamo la matrice

Corpo

```
int ff ( "  "  "  "  "  )
```

```
#define C 4
#define R 3
int ff (int m[][C], int RR, char k, int q);
int main()
{ int m[R][C] = {{2,3,4,5},{3,5,7,9},{1,2,3,4}};
  char rc;
  int pos, s
  printf ("posizione = ")
  scanf ("%d", &pos);
  printf ("riga o colonna:");
  scanf ("%*c%c", &rc);
  s = ff (m, R, rc, pos);
  printf ("s=%d", s);
  return 0;
}
int ff ( "  "  "  "  ) {
  int i, sum;
  sum = 0;
  if (k == 'r') {
    for (i=0; i<C; i++)
      sum = sum + m[q][i];
  }
  else {
    for (i=0; i<RR; i++)
      sum = sum + m[i][q];
  }
  return sum;
}
```

Nella chiamata si mette solo m come nel vettori = = indirizzo matrice

Somma riga

Somma colonna

Indica la colonna/riga di cui voglio calcolarela somma.

Bisogna inserire valori tra 0 e R-1 e o e C-1.

carattere 'r'?

NO

```
for (i=0; i<R; i++)
  sum = sum + m[i][col]
```

SI

```
for (i=0; i<C; i++)
  sum = sum + m[][]
```

m[riga][i]

[riga] = [q]

Aggiunta: devo essere sicuro di ricevere 'r' o 'c' da tastiera → do-while. Anche per la posizione devo fare in modo che vengano inseriti solo valori più bassi di quante righe e colonne ci sono.

↳ prima do-while con (rc != 'r') && (rc != 'c')
poi in base a 'c' o 'r' avrò 2 do-while con
pos < C-1
appure pos < R-1