NUMERO: 1119 DATA: 22/09/2014

A P P U N T I

STUDENTE: Caruso

MATERIA: Informatica + Eserc.

Prof. Mezzalama

LEZIONE 1 - INTRODUZIONE, CONCETTI INIZIALI

VEDI DISPENSE PER COMPENSARE

* COMPONENTE FONDAMENTALE DEL CORSO = LABORATORIO

OBIETTIVI DEL CORSO:

- PROBLEM SOLVING (AFFRONTARE E CAPIRE UN PROBLEMA ED ESSERE CAPACE DI RISOLVERLO - TIPICA SKILL INGEGNERISTICA.
- DIAGRAMMA DI FLUSSO E PSEUDO CODICE
- TERMINOLOGIA INFORMATICA CORRETTA

ANCHE SE E' UN CORSO DI PROGRAMMAZIONE, L'OBIETTIVO E' SOSTANZIALE E' IL PROBLEM SOLVING.

* CONCETTI INTRODUTTIVI

NEGLI ULTIMI 30 / 90 LA VITA QUOTIDIANA E' STATA BASATA SUL COMPUTER. INFORMATICA - DEFINIZIONE - INFORMAZIONI E DATI.

* I DATI DIGITALI: TUTTO DIVENTA **BIT**

CON L'AVVENTO DEL MONDO DIGITALE, TUTTE LE INFORMAZIONI VENIVANO CODIFICATE CON UN UNICO ALFABETO. [MENTRE PRIMA: CONTI → NUMERI; PAROLE → ALFABETO; SUONI → NOTE]. MENTRE NEL PASSATO, PER COSE DIVERSE SI USAVANO TECNOLOGIE DIVERSE, ORA LA TECNOLOGIA E' UNA SOLA PERCHE' LE INFORMAZIONI SONO TRADOTTE CON UN SOLO ALFABETO.

QUESTI **ALFABETI** SONO SEQUENZE DI BIT, DI **ZERI** E DI **UNO**.

* LE TECNOLOGIE COME FATTORE ABILITANTE DEI CAMBIAMENTI INDUSTRIALI E SOCIALI

* **DIGITALE** VS ANALOGICO = I BIT SONO I MIGLIORI PERCHE' A BASSI COSTI, HANNO UNA QUALITA' E UN'EFFICIENZA MAGGIORE. INOLTRE, LA QUALITA' RIMANE INALTERATA NEL TEMPO. [SE PRIMA FACEVI CENTO COPIE DI UNA CASSETTA AUDIO, L'AUDIO DELL'ULTIMA COPIA ERA QUASI IRRICONOSCIBILE - FACEVA SCHIFO.]. UN'ALTRO ASPETTO POSITIVO E' LA COMPRESSIONE DEI DATI E LA TRASMISSIONE EFFICIENTE DI QUESTI ULTIMI.

MEMORIZZAZIONE:

COME SI CALCOLA LA CAPACITA' DI MEMORIZZAZIONE? O MEGLIO COME FACCIO A DIMENSIONARE LA MEMORIA O UN DISCO ??

IL MODO IN CUI SI RAPPRESENTA LA CAPACITA' DI MEMORIZZAZIONE E' LA QUANTITA' DI BYTE CHE POSSO MEMORIZZARE SU SUPPORTO FISICO. DATO CHE SONO MOLTI NON PARLO DI UNITA' MA DI MIGLIAIA, MILIONI ECC.

- CHILO K - UN MIGLIAIO
- MEGA M - UN MILIONE
- GIGA G - UN MILIARDO
- TERA T - MILLE MILIARDI
- PETA P - UN MILIONE DI MILIARDI
- EXA E - UN MILIARDO DI MILIARDI

TRASPORTO

COME SI MISURA LA TRASMISSIONE? IN QUANTI BIT SI TRASMETTONO AL SECONDO. UNITA' DI MISURA: bps BIT PER SECONDO E MULTIPLI. LA CAPACITA' DI TRASMISSIONE E' FONDAMENTALE NEI SERVIZI EVOLUTI;

ESEMPIO: ADSL
↑
ASIMMETRICO (CANALI SCARICO / CARICO)

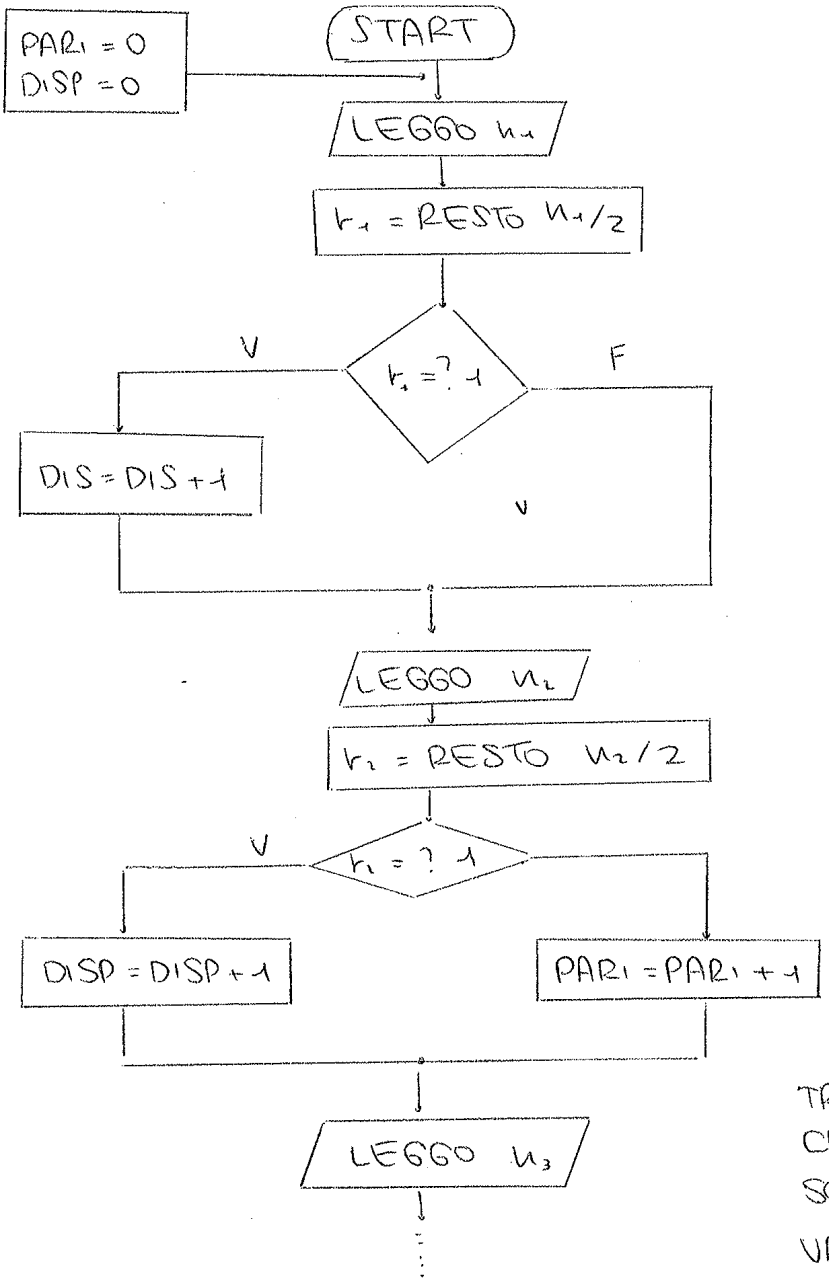
MEMORIA NEI GRANDI "PROVIDER" (AMAZON - GOOGLE): SONO IN GRADO DI GESTIRE MILIONI DI CONNESSIONI/UTENTI AL SECONDO. ESSI POSSEGGONO ANCHE MOLTA MEMORIA, QUINDI MILIONI DI MILIARDI DI BYTE IN LINEA.

STORIA = DISPENSE

PER LA TRADUZIONE → LINGUAGGI AD ALTO LIVELLO, SIMILI AL LINGUAGGIO UMANO - PER UNA QUESTIONE DI SEMPLICITA' ED ECONOMICO - INDUSTRIALE.
ESISTONO PROGRAMMI (CHIAMATI TRADUTTORI) CHE TRADUCONO IL LINGUAGGIO AD ALTO LIVELLO IN LINGUAGGIO MACCHINA

* SUPPONIAMO DI LEGGERE 10 NUMERI INTERI E DI CONTARE QUANTI SONO PARI E QUANTI DISPARI.

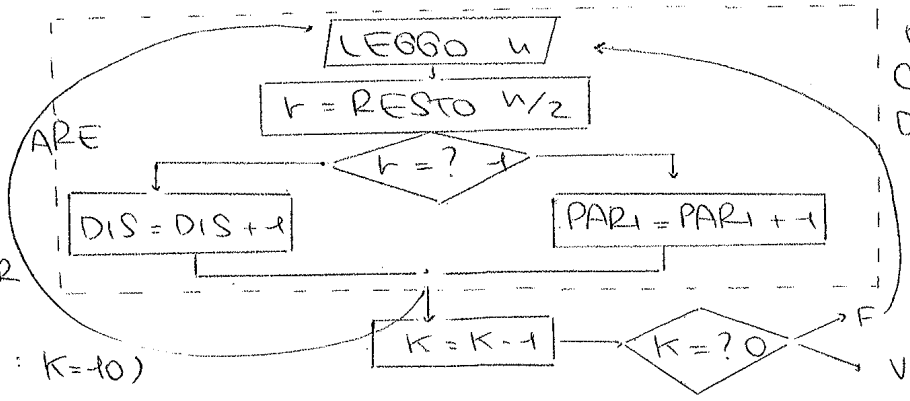
- DATI $n_1, n_2, \ldots, n_{10}$ → PARI O DISPARI
- ALGORITMO → $n/2$ → RESTO



NOTA: $r = n_1 - q*2$ QUESTA E' L'OPERAZIONE PER IL CALCOLATORE

TROVO UN ALGORITMO CHE MI EVITI SCRITTURE ECCESSIVAMENTE LUNGHE

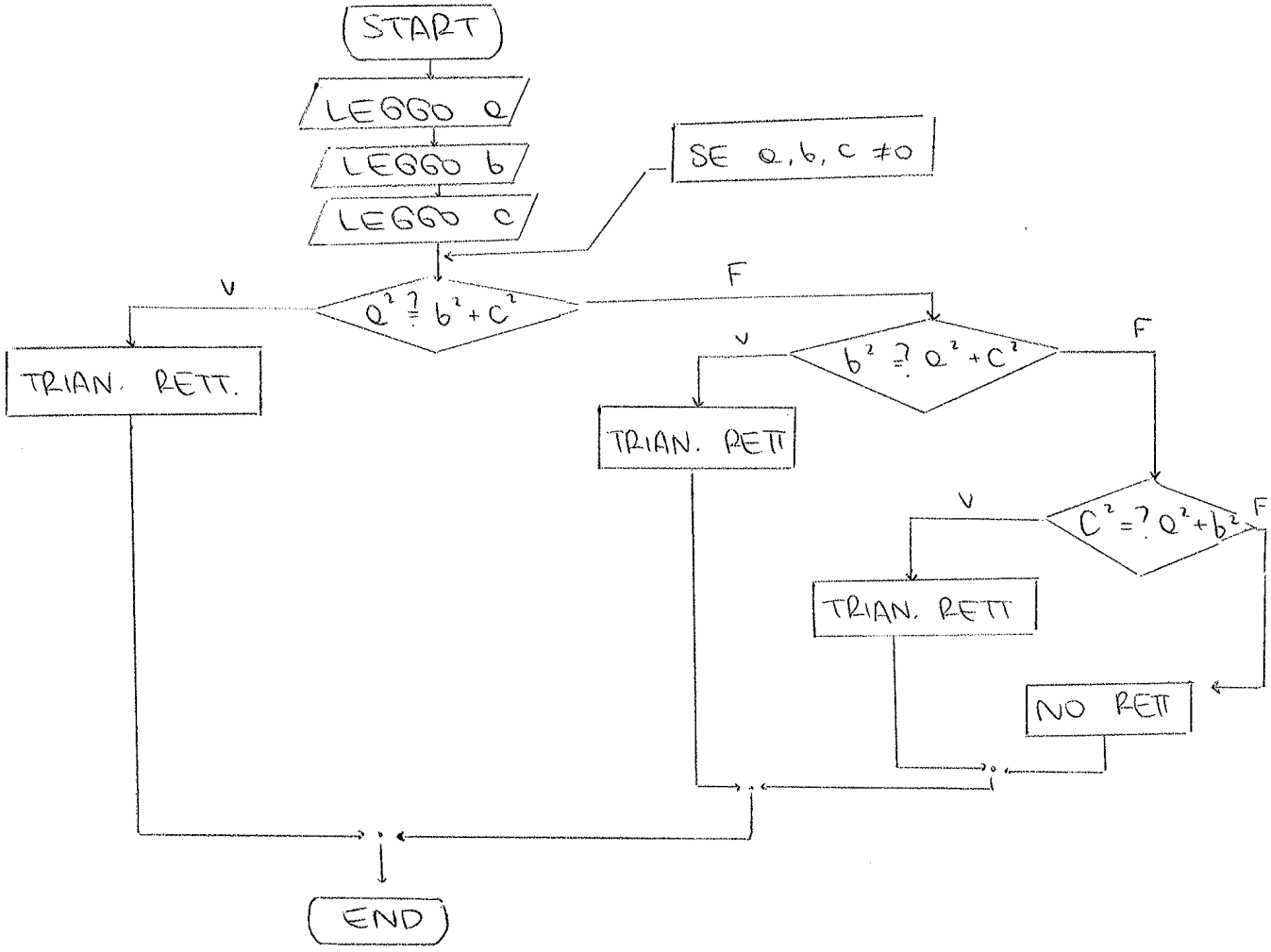
NOTA = LE OPERAZIONI SUI NUMERI SONO LE STESSE. HO QUINDI UN' UNICA OPERAZIONE CHE SI **RIPETE**. LO RISOLVO, QUINDI, COME:



DEVO RICOMINCIARE - RIPETERE L'OPERAZIONE PER K VALORI DI n (ES: K=10)

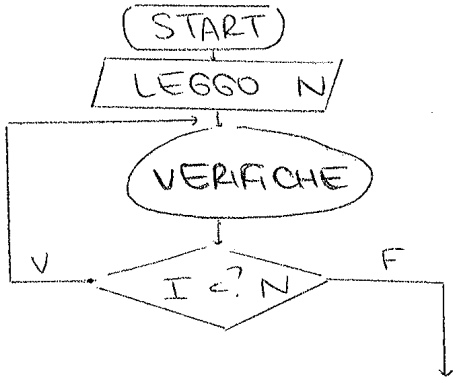
OGNI VOLTA CHE RICOMINCIA DEVE SAPERE CHE DOVRA' FARLO ANCORA K-1 VOLTE

ESERCIZIO 2 - LEGGO 3 NUMERI CORRISPONDENTI AI 3 LATI DI UN POSSIBILE TRIANGOLO. IL TRIANGOLO E' RETTANGOLO?



SPESSO, PER VERIFICARE CHE IL DIAGRAMMA DI FLUSSO SIA CORRETTO POSSO INTRODURRE DEI **NUMERI** A CASO E **VERIFICARE** CONDIZIONI E OPERAZIONI.

ESERCIZIO 3 - LEGGO UN NUMERO N DA TASTIERA, VERIFICO CHE SIA UN NUMERO PRIMO.



- QUANDO RIPETO DEVO =
  - INCREMENTARE N
  - VERIFICARE IL RESTO N/I

RISCRIVO L'ESERCIZIO IN MANIERA PIÙ PRECISA

LEZIONE 2 - ARCHITETTURA

VEDI DISPENSE

PIASTRA MADRE = RAM, CPU MICROPROCESSORE, SCHEDA RETE LAN, SCHEDA AUDIO, SCHEDA VIDEO

IN ALCUNI CALCOLATORI QUESTI ELEMENTI SONO POSIZIONATI IN MODO PERMANENTE NELLA PIASTRA M.

MEMORIA

INDIRIZZAMENTO =

LA MEMORIA E' ORGANIZZATA IN CELLE (MINIMA UNITÀ ACCESSIBILE DIRETTAMENTE). AD OGNI CELLA DI MEMORIA E' ASSOCIATO UN INDIRIZZO (NUMERICO) PER IDENTIFICARLA IN MODO UNIVOCO.

IL CALCOLATORE E' MOLTO VELOCE, (VELOCITA' DI GHz), ALLORA ANCHE LA MEMORIA DEVE ESSERE TALE.

MEMORIA **INTERNA** = SI TROVA ALL'INTERNO DELL'ELABORATORE ED E' ALLO STATO SOLIDO (CHIP). NE ESISTONO DUE TIPI:

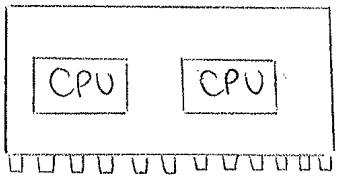
- RAM = MEMORIA VOLATILE, OSSIA UNA VOLTA SPENTO IL CALCOLATORE PERDO I DATI. E' VELOCE ED E' LA MEMORIA CHE DIALOGA CON IL PROCESSORE.
- ROM = HANNO UN VANTAGGIO → L'INFORMAZIONE RESTA PERMANENTE

* VELOCITA' = NANO SECONDI
* QUANTITA' LIMITATA (QUALCHE GB)
* NON RIMOVIBILE
* COSTOSA (0.1 €/MB)

MEMORIA **ESTERNA** = LEGATA A TECNOLOGIE NON ELETTRONICHE, MA OTTICHE E MAGNETICHE. E' PERMANENTE, ANCHE SE PIÙ LENTA DELLA MEMORIA INTERNA

RAM =
* CIRCUITI INTEGRATI
* IL TEMPO DI ACCESSO E' COSTANTE
* T_a = COSTANTE
* MEMORIA INTERNA VOLATILE

ESEMPIO: CPU DUAL-CORE



DENTRO AL CHIP, SE CI STANNO, CI METTO 2 CPU. ← VENGONO CHIAMATE "CORE"

ES: INTEL i7

* ULTIMA GENERAZIONE DI MICROPROCESSORI: MULTI-CORE. [DA 4 A 6 CORE]
* ARCHITETTURA 64 BIT (BUS)
* PROCESSO DI FABBRICAZIONE: VI SONO DELLE MACCHINE CON UNA PRECISIONE DI 20-45 NANO METRI CHE INCIDONO IL SILICIO

DATI (VEDI DISPENSE)

QUANDO SCRIVO UN PROGRAMMA DEVO FARE DISTINZIONE TRA DUE PARTI: UNA DICHIARATIVA E UNA ESECUTIVA. NELLA PARTE DICHIARATIVO DEVO METTERE I NOMI DELLE VARIABILI CHE VOGLIO ADOPERARE [OSSIA LI' SOSTANZIALMENTE CI METTO I DATI]; I DATI POI VENGONO ADOPERATI NELLA FASE ESECUTIVA, GENERALMENTE COMPOSTA DA UNA SERIE DI ISTRUZIONI

NUMERI REALI

0 1 0 1 . 1 1

$2^3\ 2^2\ 2^1\ 2^0\ .\ 2^{-1}\ 2^{-2}$ → $4 + 1 + \frac{1}{2} + \frac{1}{4} = 5,75$

↑

E' LA VIRGOLA

QUESTO METODO 1, PER QUESTIONI DI VELOCITA', NON VIENE MOLTO UTILIZZATO. SI PREFERISCE, INFATTI, UN METODO CHIAMATO FLOATING POINT.

ESERCIZIO:

1 1 1 1 1 0 0 1 1 0 — RAPPRESENTANO UN NUMERO IN FLOATING POINT

± | E | M

PRIMO BIT DI SEGNO = 1 => SEGNO NEGATIVO

NUMERO = $-(0\ 0\ 1\ 1\ 0) \cdot 2^{(1\ 1\ 1\ 1)} = -6 \cdot 2^{(1\ 1\ 1\ 1)}$

M | E

1 1 1 1 => NUMERO RELATIVO, OVVIAMENTE SCRITTO IN COMPLEMENTO A 2. POSSO RICAVARE CHE NUMERO E' CON I DUE METODI DI PRIMA

1 1 1 1 = -1 => $-6 \cdot 2^{-1} = -6 \cdot \frac{1}{2} = -3.0$

* ANCHE IL COMPUTER NON E' PERFETTO!

INNANZITUTTO PERCHE' NELL'UTILIZZO DEI NUMERI ESSO ADOTTA SEMPRE DEI SISTEMI POSIZIONALI, OSSIA LI RAPPRESENTA COME SEQUENZA FINITA DI CIFRE.

AD ESEMPIO, NOI UTILIZZIAMO DELLE FRAZIONI CHE CI PERMETTONO DI SCRIVERE NUMERI INFINITI UTILIZZANDO DEI NUMERI FINITI. QUESTO TIPO DI COSE NON SONO ADOPERATE DAL CALCOLATORE.

IL CALCOLATORE, AL CONTRARIO, APPROSSIMA I NUMERI INFINITI PER CONSIDERARLI FINITI. TALE APPROSSIMAZIONE COMPORTA UNA LIEVE MANCANZA DI PRECISIONE.

ALTRE IMPERFEZIONI SONO:

- OVERFLOW
- L'INTERVALLO DI RAPPRESENTAZIONE

CODIFICA DEI TESTI

VEDI DISPENSE

QUANDO AD ESEMPIO SI REALIZZA UN CARTONE ANIMATO, CIO' CHE SI FA' NON E' PORTARE ALL'INTERNO DEL CALCOLATORE DELLE IMMAGINI GIA ESISTENTI [CHE E' QUELLO CHE FA UNA MACCHINA FOTOGRAFICA O UNO SCANNER]. AL CONTRARIO, SI COSTRUISCONO DELLE IMMAGINI E PER FAR CIO' SI NECESSITA L'UTILIZZO DELLA GEOMETRIA [PIANA, SOLIDA ECC.]. QUESTE COMPONENTI GEOMETRICI SONO MOLTO PICCOLI E BEN COMBINATI POSSONO CREARE L'IMMAGINE.

QUELLO CHE IL CALCOLATORE MEMORIZZA E' L'INFORMAZIONE GEOMETRICA. SI APPROSSIMA UN REALTA' IN 3D. IL FILE MEMORIZZATO E' UN FILE GEOMETRICO, PER TALE MOTIVO QUESTE TIPO DI **RAPPRESENTAZIONI** VENGONO CHIAMATE **VETTORIALI**.

MONDO REALE:

FOTOGRAFIA / SCANSIONE → IMMAGINE E PIXEL → SCHERMATA COMPUTER O TV

MONDO VIRTUALE:

GEOMETRIA SPAZIO 3D VETTORI → SOLIDI O FIGURE GEOMETRICHE → IMMAGINE PIXEL 2D → SCHERMO

DATA COMPRESSION:

IL LATO POSITIVO DELLA CODIFICA IN BIT E' LA FACOLTA' DI COMPRIMERE I DATI. CON IL TERMINE "COMPRESSIONE DEI DATI SI FA RIFERIMENTO AD UN INSIEME DI METODI CHE HANNO COME OBIETTIVO LA RIDUZIONE DEL NUMERO DI BIT NECESSARI PER IMMAGAZZINARE UN'INFORMAZIONE, GENERALMENTE UN FILE.

CI SONO DUE TIPI DI COMPRESSIONE:

* SENZA PERDITA (**LOSELESS**): CODIFICA DIVERSA DEI BIT IN MODO CHE SE VOGLIO INDIETRO IL FILE ORIGINALE DA QUELLO COMPRESSO, POSSO RECUPERARE TUTTI GLI ELEMENTI INIZIALI.
* CON PERDITA (**LOSSY**): I PIXEL, DURANTE LA COMPRESSIONE VENGONO, CANCELLATI ANCHE SE L'UTENTE NON SE NE ACCORGE.

COMPRESSIONE DEI FILMATI

AL FINE DI RIDURRE L'INFORMAZIONE TRASMESSA, PER RIDURRE LA BANDA PER CANALE, SI APPLICANO NELLE TRASMISSIONI DIGITALI TECNICHE DI COMPRESSIONE CON PERDITA. ALCUNI FRAME SI CODIFICANO TUTTI, ALTRI SOLO NELLE PARTI CHE DIFFERISCONO DA QUELLI ADIACENTI.

SOSTANZIALMENTE, NELLA COMPRESSIONE:

- SI PRENDE LA PRIMA IMMAGINE E LA SI CODIFICA IN JPEG
- POI SI PRENDE LA SECONDA IMMAGINE E SI TRASMETTONO SOLO LE VARIAZIONI
- DOPO CHE LE VARIAZIONI SONO TROPPE CAMBIO FRAME

QUESTA TECNICA VIENE CHIAMATA **MPEG** [MOVING PICTURE EXPERTS GROUP], INVENTATA DAGLI ITALIANI, UTILIZZATO PIU' DI TUTTI NEL MONDO DELLA TELEVISIONE.

INOLTRE, NEL MPEG 2 NON SONO COMPRESSE SOLO LE IMMAGINI MA ANCHE I SUONI.

L'ALGORITMO CHE CODIFICA IL **SUONO** E' DEL TIPO .mp3

CODIFICA DEL SUONO

QUANDO PARLIAMO DELLA CODIFICA DEI SUONI E DELLA VOCE. DOBBIAMO FARE RIFERIMENTO A UNA CLASSICA SEQUENZA. SE PRENDO AD ESEMPIO UN REGISTRATORE, LA SEQUENZA E':

VOCE [FENOMENO FISICO LEGATO A ONDE DI PRESSIONE] → MICROFONO (TRADUTTORE) [TRASFORMA LE ONDE DI PRESSIONE IN SEGNALE ELETTRICO] → CONVERSORE ANALOGICO/DIGITALE

GRANDEZZA FISICA → TRADUTTORE → GRAND. FISICA SEGNALE ELETT. → CONVERSORE → BIT

IL CONVERSORE ANALOGICO / DIGITALE TRASFORMA IL SEGNALE ELETTRICO IN UNA SEQUENZA DI BIT. LA TRADUZIONE IN BIT AVVIENE ATTRAVERSO UN MECCANISMO CHIAMATO **CAMPIONAMENTO**, OSSIA, DATO UN ANDAMENTO [CHE RAPPRESENTA IL SEGNALE ELETTRICO] RAPPRESENTA TALE ANDAMENTO DEFINENDO AD INTERVALLI REGOLARI IL VALORE DELL'ANDAMENTO STESSO. LA FREQUENZA CON CUI VADO A LEGGERE QUESTI INTERVALLI E' DETTA DI CAMPIONAMENTO ED E' MISURATA IN HERTZ.

LEZIONE 10 - INTRODUZIONE AL C E ISTRUZIONI ELEM.

NOTA : SE IO AVESSI LA POSSIBILITA' DI GUARDARE NELLA MEMORIA E VEDESSI UNA **SEQUENZA** DEL TIPO 01101 IO NON POSSO SAPERE CHE **TIPO DI INFORMAZIONE** E' QUELLA [ASCII, NUM. INTERO, RELATIVO O REALE, PARTE DI UN NUMERO, CARATTERE DI CONTROLLO]. AL CONTRARIO, LO SA IL CALCOLATORE IN BASE ALLA POSIZIONE DI QUEL CODICE. IL CALCOLATORE, INFATTI, DISPONE ALL'INTERNO DELLA MEMORIA LE INFORMAZIONI E QUINDI LE SEQUENZE DI CODICI BINARI IN **POST.** BEN PRECISI.

NOTA 2 : SE HO UNA SEQUENZA IN COMPLEMENTO A 2 NELLA SOMMA AVVIENE L'OVERFLOW. NELL'HARDWARE C'E' QUALCOSA CHE VERIFICA L'OVERFLOW, BASANDOSI SUI SEGNI DEGLI OPERANDI.

NOTA 3 : [S] [E] [M] NEL CALCOLATORI CI SONO DUE PALLOTTOLIERI E UNO E' PER I NUMERI REALI, CHE ESEGUE CALCOLI COMPLICATI [FPU]

STRUTTURA DEL PROGRAMMA

```
PARTE DICHIARATIVA
main ()
{
   PARTE DIC. LOCALE
   PARTE ESECUTIVA
   return 0;
}
```

— DOVE DEFINISCO LE "SCATOLE" SCRIVENDO IL TIPO DI SCATOLA CHE USO ES. int A - float b

* DEVO SEMPRE CHIUDERE LE FRASI CON ;
  * ASSEGNAZIONE
  * ISTRUZIONE INPUT / OUTPUT
  * PRINTF E SCANF
  * INCLUDE E DEFINE

  → VEDI **DISPENSE**

  * OPERATORI ARITMETICI - RELAZIONALI
    DI CONFRONTO -
    DI INCREMENTO

* ESERCIZIO DA ESAME

f_1 f_2 SONO 2 FUNZIONI DI 3 VARIABILI x y z

$f_1 = (\overline{x + y}) + z$ $f_2 = x + y\bar{z}$ SONO UGUALI ?

* DEVO COSTRUIRE LE TABELLE DELLA VERITA' DI f_1 E f_2
* VERIFICARE CHE, A PARI CONDIZIONI D'INGRESSO, LE DUE FUNZIONI SIANO UGUALI

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

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

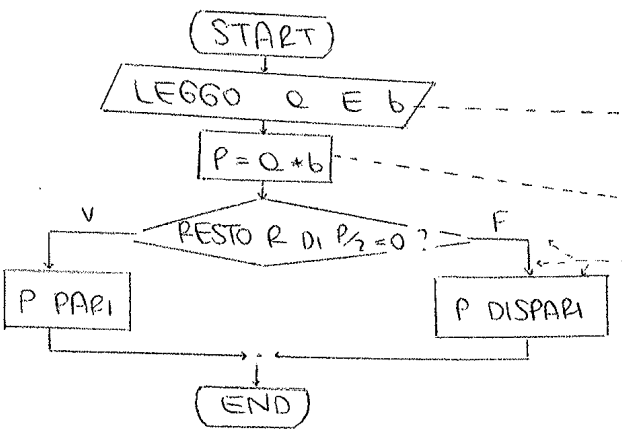
LE DUE FUNZIONI NON SONO UGUALI PERCHE' HO VERIFICATO CHE A FRONTE DELLA PRIMA COMBINAZIONE $f_1 = 1$ MENTRE $f_2 = 0$

L'ALGEBRA BOOLEANA SI APPLICA IN DUE CONTESTI:
* LA REALIZZAZIONE DEL CIRCUITO
* LA PROGRAMMAZIONE

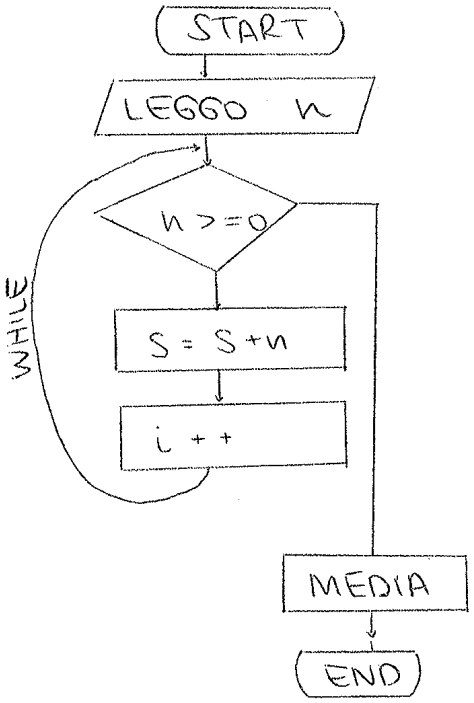
OPERATORI BOOLEANI: OR NOT AND
AND => && OR => ||

ESEMPIO: if(( ) AND( ))
↑ LE CONDIZIONI SI DEVONO ATTUARE CONTEMPORANEAMENTE

* 1° ESERCIZIO - ISTRUZIONE IF
LEGGO 2 NUMERI DA TASTIERA E VERIFICARE SE IL PRODOTTO E' PARI.




```
int main()
{ int a, b, P;
 printf("1 NUM:"); scanf("%d", &a);
 printf("2 NUM:"); scanf("%d", &b);
 P = a*b;
 if(P%2 =0)                ← ESPRESSIONE CHE CALCOLA IL RESTO
   {printf("PARI");}
 else
   {printf("DISPARI");};
 return 0; }
```



FLOW - CHART

→ QUESTA ISTRUZIONE MI SERVE SOLO PER LA PRIMA VOLTA

→ CONDIZIONE DEL WHILE

→ BLOCCO / AZIONE

→ INCREMENTO DELLA VARIABILE DI CICLO

] ISTRUZIONE DEL WHILE

DATI

* MEDIA float
* S, N, i int

PROGRAMMA

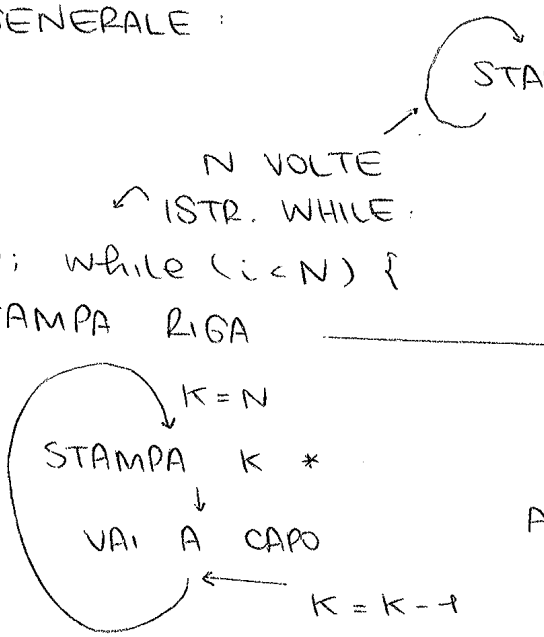
```
# include <stdio.h>
# include <stdLib.h>
int main() { int S, n, i; float MEDIA;
        i = 0;   S = 0;
        Printf("Dato = ");
        Scanf("%d", &n); if(n<0) {printf("NO MEDIA");}
        else {
        While (n >= 0) { S = S+n; i++;
                Printf ("Dato = ");
                Scanf("%d, &n); }
        MEDIA = S / i;
                (float)
                Printf("% f \n", MEDIA); }
        return 0;
}
```

NEL CASO IN CUI IL PRIMO NUMERO SIA < 0 NON POSSO FARE LA MEDIA

ISTRUZIONE "VOGLIO CHE CALCOLI LA MEDIA COME UN DECIMALE

* ESERCIZIO "CICLI ANNIDATI". STAMPARE UN TRIANGOLO RETT. DI LATO N. CON N ASTERISCHI SU UN CATETO E N RIGHE.

IN GENERALE:



DEVO USARE 2 WHILE

SIGNIFICA STAMPARE K ASTERISCHI:

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

ALL'INIZIO K=N
AVANZANDO CON I CICLI
K DIMINUISCE

PROGRAMMA:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{ int i, J, K;
    i=0;
    K=0;
while (i<N) {  J=0;
    while (J<K) {
    printf("*");
    J++;
    }
printf("\n");
K=K-1;
i++;
}
return 0;}
```

```
#define N 7
```

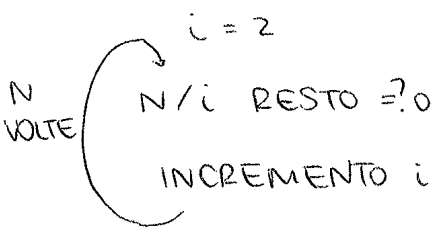
DEVO RICORDARMI DI INIZIALIZZARE LE VARIABILI PRIMA DEL CICLO. INOLTRE, NEL CASO DEI CICLI ANNIDATI, CONVIENE SEMPRE PARTIRE DA QUELLO PIU' INTERNO

LEZIONE 16 - ESERCIZI E VETTORI

* **ESERCIZIO** - CREARE UN PROGRAMMA CHE, INTRODOTTO UN NUMERO, VERIFICHI SE E' PRIMO

• **ANALISI DELLA TRACCIA**: COME FACCIO A SAPERE CHE UN NUMERO E' PRIMO? SE DIVISIBILE SOLO PER 1 E PER SE STESSO

• **COSA FACCIO?** DIVIDO IL NUMERO PER TUTTI I NUMERI MINORI DI ESSO E NOTO SE TALE DIVISIONE PORTA A UN RESTO UGUALE O DIVERSO DA ZERO



POSSO ADOPERARE IL FOR DATO CHE CONOSCO IL NUMERO DI VOLTE CHE RIPETO IL CICLO

N = NUMERO CHE LEGGO DA TASTIERA

i = VARIABILE DI CICLO

FLAG = BANDIERINA [ALZATA = 1 = NUM. PRIMO / ABBASSATA = 0 = NO]

NOTA => NON POSSO INIZIALIZZARE i = 0 PERCHE' POI NON POTREI FARE N/i; NE' MI CONVIENE PORRE i = 1 IN QUANTO NON MI AIUTA AL FINE DELLA RISOLUZIONE DELL'ESERCIZIO

" = " SIMBOLO DELL'ASSEGNAZIONE IN INFORMATICA

" = = " SIMBOLO DELL'UGUALE COME INTESO IN MATEMATICA

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int main()
5   { int N, i, FLAG, r;
6   printf("Numero =");
7   scanf("%d", &N);
8   FLAG=0;
9       for(i=2;i<N;i++) {
10      r=N%i;
11      if(r==0)
12      FLAG=1;
13          }
14  if(FLAG==1)
15  printf("No primo");
16  else
17  printf("Primo");
18
19      return 0;
20  }
21
```

LEZIONE 18 - ESERCIZI SUI VETTORI

* **ESERCIZIO** - SUPPONGO CHE IL CALCOLATORE, MEDIANTE UN PROGRAMMA, DEBBA GESTIRE LA PIOVOSITA' NELL'ARCO DI UNA SETTIMANA TRAMITE UN SENSORE. IL SENSORE VIENE "LETTO" UNA VOLTA AL GIORNO, AL TERMINE DI QUEST'ULTIMO.

- **PROGRAMMA** DEVE:
  * LEGGERE DA TASTIERA I VALORI DI PIOVOSITA'
  * CALCOLI LA PIOVOSITA' MEDIA DELLA SETTIMANA
  * CALCOLI QUANDO LA PIOVOSITA' GIORNALIERA E' SUPERIORE AD UNA DETERMINATA SOGLIA [VALORE IMMESSO DA TASTIERA]
  * CALCOLARE LA DIFFERENZA MASSIMA DI PIOVOSITA'.

LA STRUTTURA DATI ADOPERATA SARA' UN **VETTORE**, COSTITUITO DA SETTE ELEMENTI, OSSIA NUMERI REALI.

- float v[N] ← LEGGO DA TASTIERA OPERANDO IL for
- CALCOLO DELLA MEDIA: PER FARE LA SOMMA DEGLI ELEMENTI DI UN VETTORE DEVO USARE LA STRUTTURA ITERATIVA SOMMA = SOMMA + V_i. OPERO NUOVAMENTE IL for IN MODO DA SOMMARE TUTTI GLI ELEMENTI.
- NEL DOVER CALCOLARE LA DIFFERENZA MASSIMA DI PIOVOSITA' DEVO TROVARE IL VETTORE CON VALORE MASSIMO E IL VETTORE CON VALORE MINIMO. PER INIZIALIZZARE IL MASSIMO LO PONGO UGUALE AL PRIMO ELEMENTO [ANCHE PER IL MINIMO].

COME FACCIO A CALCOLARE IL MAX E AD ADOPERARE LO SPOSTAMENTO?

MAX => V[k] - V[N-1] : PRENDO IL PRIMO ELEMENTO E LO CONFRONTO CON UNO DI RIFERIMENTO. SE E' MAGGIORE, LO CONFRONTO CON IL SECONDO ELEMENTO E COSI' VIA. SE E' MINORE CONFRONTO IL SECONDO ELEMENTO CON IL TERZO ECC... = for(i=k; i<N-1; i++)

{ if(V[i] > MAX)

max = V[i];

(DEVO SEMPRE DEFINIRE (INIZIALIZZARE) IL VALORE DI MAX. max = 0 OPPURE max = 1° ELEMENTO)

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define N 5
int main()
{ int v[N], k, i, max, pmax, temp;
max=0;
/*invece di inserirli da tastiera, i valori di N li prendo in maniera casuale*/
srand(time(NULL));
  for(i=0;i<N;i++)
  v[i]=rand()%100+1;
/*questa operazione mi associato a un generico elemento v[i] un valore a caso RAND -
in questo modo ho "popolato" il mio vettore*/
/* simica esercizio da esame "leggere da tastiera una serie di numeri chè si trovino
in un intervallo predefinito"*/
for(i=0;i<N;i++)
printf("%d\n", v[i]);

for(k=0;k<N-1;k++)
{  /*calcolo il max*/
   for(i=k;i<N-i;i++)
       {
       if(v[i]>max)
       {max=v[i]; pmax=i;}
       }
}
/*scostamento*/
temp=v[k];
v[k]=v[pmax];
v[pmax]=temp;

return 0;
}
```

↑ ESERCIZIO 1

ESERCIZIO 2 →

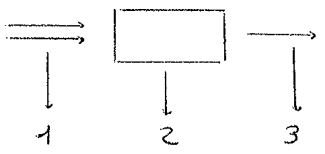
```c
#include <stdio.h>
#include <stdlib.h>
#define N 6
int main()
{ int v[N], i, neg, n;
  v[N]=0;
  printf("Inserisci n=");
  scanf("%d", &n);
    if (n<0){
    neg=1;
    n=-n-1; }
    else {
    neg =0; }
         for(i=0;i<N;i++)
         {
            if(neg==1){
            v[i]=1-n%2;}
            else {
            v[i]=n%2; }
            n=n/2;
         }
  for(i=0;i<N;i++){
  printf("%d", v[i]);}

  return 0;
}
```

LEZIONE 20 - FUNZIONI

INTRODUZIONE, DEFINIZIONE → [VEDI DISPENSE]

* PARTE "DICHIARATIVA"

ALL' INIZIO DEL PROGRAMMA DEVO DICHIARARE 3 ELEMENTI:



1. QUANTI E QUALI SONO I PARAMETRI D' INGRESSO
2. NOME
3. TIPO

SINTASSI - ORDINE IN CUI SI SCRIVONO:

< tipo > < nome > (< parametri >)

INOLTRE, PER OGNI PARAMETRO DEVO INDICARE [ADOPERARE] UN NOME E UN TIPO. ESEMPIO =

```
int  cubo (int a);   ←  DICHIARAZIONE
TIPO NOME  PARAMETRO
           TIPO NOME
```

* PARTE "**ADOPERARE**"

QUANDO ADOPERO UNA FUNZIONE / MODULO, INNANZITUTTO, DEVO TENER PRESENTE LA SINTASSI: `u = cubo(x);` IN MODO CHE NEL PROGRAMMA ASSOCIO ALLA VARIABILE LA FUNZIONE

* PARTE "**DEFINIZIONE o CORPO**"

NEL PROGRAMMA AUTOCONSISTENTE [QUESTA PARTE] DEVO SCRIVERE LE ISTRUZIONI CHE REALIZZANO LA FUNZIONE DEL MODULO [AL FONDO DEL `main`]

SINTASSI: **1° REGOLA** = DEVE INIZIARE CON UNA SEQUENZA SIMILE A QUELLA FATTA NELLA FASE DELLA DICHIARAZIONE.

2° REGOLA = DEVE FINIRE CON UN `return < valore >`. QUESTO VALORE PUO' ESSERE O UNA COSTANTE O UNA VARIABILE

* DEVO CREARE LA FUNZIONE:
* DEVO COSTRUIRE IL CORPO DELLA FUNZIONE:

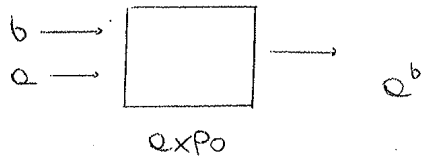
```
int leggi (int min, int max) { int a;
do  printf(" numero=");
    scanf("%d", &a);
while (a<min)||(a>max)
    /* SE LA CONDIZIONE DEL
     WHILE E' GIUSTA IL
     PROGRAMMA TORNA AL do
    (NUMERO CHE NON MI
     SERVE); ALTRIMENTI IL
     NUMERO (COMPRESO NEL
    MIO INTERVALLO) - INSCA-
    TOLATO IL a DEVO
    RITORNARLO FUORI DAL
    MODULO */
return a; }
```

NOTA = QUANDO DEFINISCO UNA VARIABILE DENTRO LA FUNZIONE, TALE VAR. E' UNA SCATOLA ADOPERATA ALL'INTERNO DELLA FUNZIONE, DEVE ESSERE DICHIARATA ANCHE DENTRO LA FUNZ.

NOTA - CONCLUSIONE:

* TUTTE LE VARIABILI ADOPERATE DEVONO ESSERE DICHIARATE DENTRO LA FUNZIONE
* PER FAR TORNARE UN RISULTATO / VARIABILE AL PROGRAMMA PRINCIPALE [FUORI DALLA FUNZIONE] DEVO USARE return <NOME DELLA VARIABILE>.

* ESERCIZIO - SCRIVERE UNA FUNZIONE CHE CALCOLI a^b




```
1   #include <stdio.h>
2   #include <stdlib.h>
3   int expo (int a, int b);
4   int main()
5   {  int n, n1, n2;
6    printf("n1=");
7    scanf("%d", &n1);
8    printf("n2=");
9    scanf("%d", &n2);
10    n=expo(n1,n2);
11    printf("expo= %d", n);
12  return 0;
13  }
14  int expo (int a, int b) {
15  int pr, i;
16  pr=1;
17  for(i=0;i<b;i++)
18      pr=pr*a;
19  return pr;
20  }
```

LEZIONE 22 - ESERCIZI SU CARATTERI

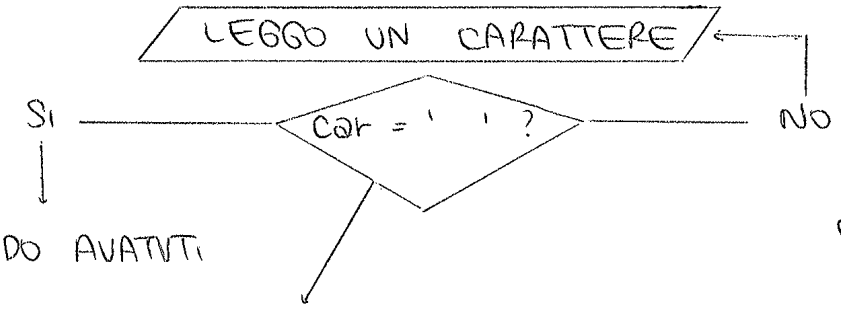
* **ESERCIZIO 1** - DATI DUE VETTORI DI CARATTERI, DI CUI UNO DEFINITO E L'ALTRO LETTO DA TASTIERA C_1 C_2. VOGLIO CALCOLARE LE LETTERE DI C_2 PRESENTI IN C_1 [INDIPENDENTEMENTE DAL MAIUSCOLO / MINUSCOLO].

  * **DATI** = char c1[N] = { SEQ. DEGLI ELEMENTI } ↙ SCRITTI COME "u"

    char c2[N] = ... LO LASCIAMO INDEFINITO

* **OPERAZIONE 1** : DEFINIRE C1 - LEGGERE C2 = DEVE TERMINARE QUANDO INTRODUCO UNO SPAZIO

PARTE DI PROGRAMMA



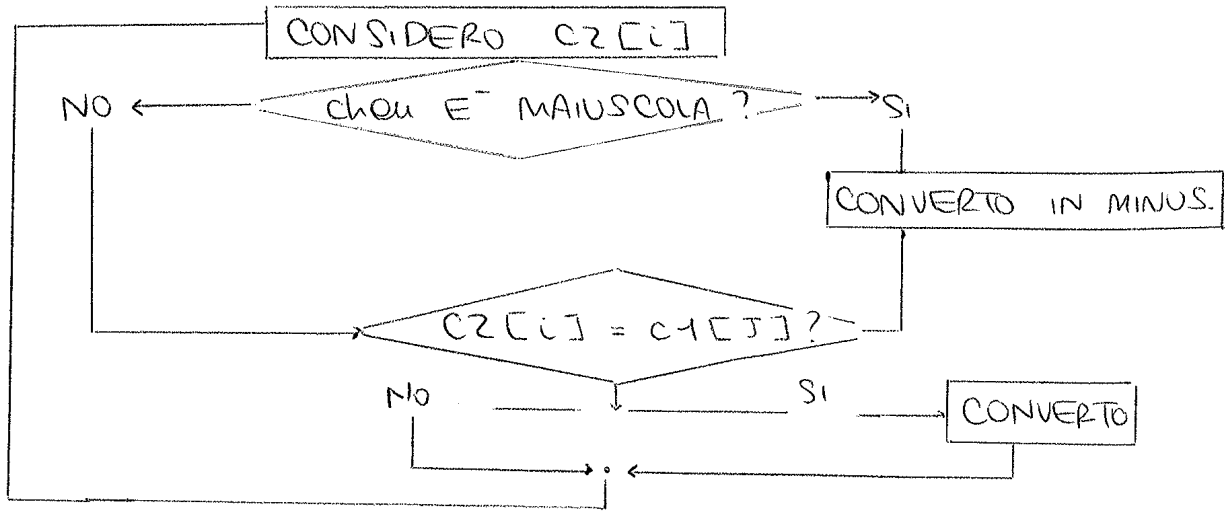
PIU' PRECISAMENTE DOVRA' ESSERE UNA CONDIZIONE BOOLEANA

(i < N) AND (c2[i-1] = '_') PERCHE' SE SUPERO N = 5 NON SO DOVE VADO A SCRIVERE IN MEMORIA

* **OPERAZIONE 2** : AVENDO ▭ C1 - N - TUTTI MINUSCOLI

▭ C2 - NN

DEVO CONFRONTARE I CARATTERI DEI VETTORI, INDIPENDENTEMENTE CHE SIANO MAIUSCOLE O MINUSCOLE. AVENDO IPOTIZZATO CHE TUTTI GLI ELEMENTI DI C1 SONO MINUSCOLI:



LEZIONE 23 - STRINGHE

CONSIDERAZIONI RIGUARDO LE **STRINGHE**:

* QUANDO LE **DICHIARO**, DEFINISCO char s[N+1] RISERVANDO UN CARATTERE AL CODICE DI FINE STRINGA;

* **FATTORE DI CONVERSIONE** = %s ES: scanf("%s", NOME), MENTRE I VETTORI NON POSSONO ESSERE CONSIDERATI COME UN CORPO UNICO, LE STRINGHE SI; CON LE STRINGHE NON ADOPERO L'& COMMERCIALE.

* LA **MANIPOLAZIONE** DELLE STRINGHE PUO' ESSERE FATTA IN DUE MODI:

- CON LE FUNZIONI S1 - S2, UTILIZZO stringcompare (S1, S2) - [ABBREVIATO strcmp]
- OPERANDO COME CON DEI VETTORI DI CARATTERI ED E' COSI' POSSIBILE TRATTARE I SINGOLI ELEMENTI. - UTILIZZO IL CONFRONTO TRA VETTORI [ELEMENTO PER ELEMENTO]

* LA **LETTURA** DELLE STRINGHE PUO' ESSERE FATTA IN DUE MODI:

- scanf("%s", __)
- get (__)

* LA **SCRITTURA** DI UNA STRINGA SI ATTUA CON:

- printf("%s" __)
- puts(__)

* ESERCIZIO 1 - LEGGERE DA TASTIERA UNA STRINGA E CALCOLARE QUANTI SONO GLI ELEMENTI CORRISPONDENTI A CIFRE. INOLTRE TRADURRE TUTTI GLI ELEMENTI DELLA STRINGA DA MAIUSCOLE IN MINUSCOLE.

LEZIONE 23 - STRINGHE

CONSIDERAZIONI RIGUARDO LE **STRINGHE**:

- QUANDO LE **DICHIARO**, DEFINISCO char s[N+1] RISERVANDO UN CARATTERE AL CODICE DI FINE STRINGA;
- **FATTORE DI CONVERSIONE** = %s ES: scanf("%s", NOME)

MENTRE I VETTORI NON POSSONO ESSERE CONSIDERATI COME UN CORPO UNICO, LE STRINGHE SI; CON LE STRINGHE NON ADOPERO L'& COMMERCIALE.

* LA **MANIPOLAZIONE** DELLE STRINGHE PUO' ESSERE FATTA IN DUE MODI:
  - CON LE FUNZIONI S1 - S2, UTILIZZO stringcompare (S1, S2) - [ABBREVIATO strcmp]

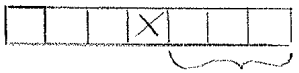
    OPERANDO COME CON DEI VETTORI DI CARATTERI ED E' COSI' POSSIBILE TRATTARE I SINGOLI ELEMENTI. - UTILIZZO IL CONFRONTRO TRA VETTORI [ELEMENTO PER ELEMENTO]

* LA **LETTURA** DELLE STRINGHE PUO' ESSERE FATTA IN DUE MODI:
  - scanf("%s", __)
  - get (__)

* LA **SCRITTURA** DI UNA STRINGA SI ATTUA CON:
  - printf("%s" __)
  - puts(__)

* ESERCIZIO 1 - LEGGERE DA TASTIERA UNA STRINGA E CALCOLARE QUANTI SONO GLI ELEMENTI CORRISPONDENTI A CIFRE. INOLTRE TRADURRE TUTTI GLI ELEMENTI DELLA STRINGA DA MAIUSCOLE IN MINUSCOLE.

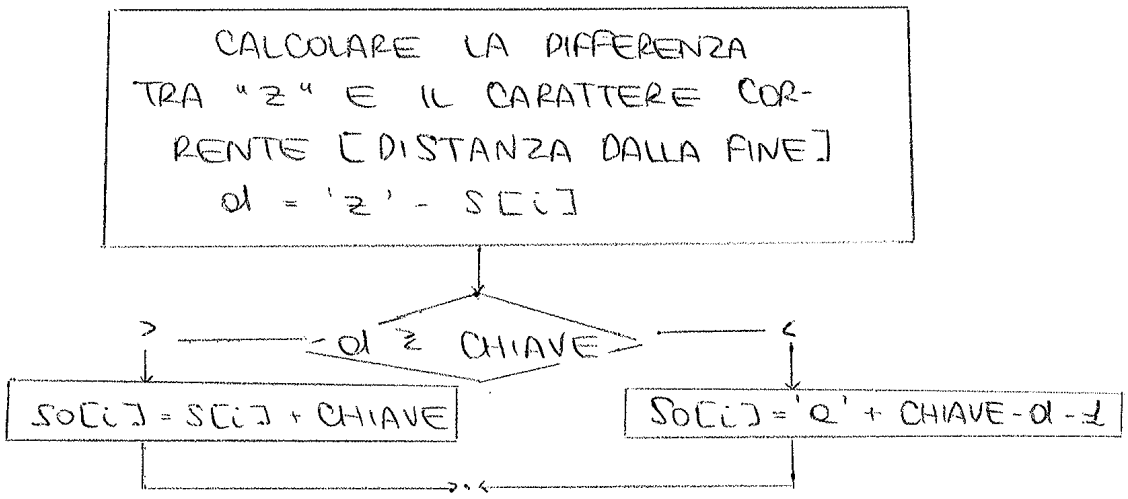
CONDIZIONE ESTREMA: SE HO LA Z, L'ASCII NON FUNZIONA COME ALFABETO CIRCOLARE ← DEVO GENERARE PERCIO' UN CICLO



SE SONO IN X COME FACCIO A SAPERE QUANTO VALE ~~ ?

CON IL CODICE ASCII POSSO FARE Z - X = E OTTENERE UN CODICE CHE MI INDICA QUANTO VALE ~~.

ALGORITMO:




```c
#include <stdio.h>
#include <stdlib.h>
#define N 5
int main()
{ char s[N]={'m','a','z','c','o'},so[N];
  int ch, d, i;
  printf("Chiave=");
  scanf("%d", &ch);
  for(i=0;i<N;i++)
  {
    d='z'-s[i];
    if(ch>d)
        so[i]='a'+(ch-d)-1;
    else
        so[i]=s[i]+ch;
  }
  for(i=0;i<N;i++){
    printf("%c, so[i]");}
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{ char c1,c2,s[70];
/*in realtà i caratteri di s sono 69 perchè
c'è il carattere di fine stringa*/
int cont, i;
printf("carattere=");
scanf("%c%*c", &c1);
printf("carattere=");
scanf("%c%*c", &c2);
/* ATTENZIONE al carattere "a capo"*/
printf("stringa=");
scanf("%s", s);
/*stringa = senza &*/
cont=0;
for(i=0;i<(strlen(s)-1);i++){
    if((s[i]==c1)&&(s[i]+1==c2)){
        cont++;} }
printf("coppia c1,c2 compare %d volte", cont);
    return 0;
}
```

* ESERCIZIO 2 - LEGGERE DA TASTIERA DELLE STRINGHE, CHE SONO NEL SEGUENTE FORMATO <nome> ␣ {U, D} ␣ <peso>. DEVO CALCOLARE IL MAX DI U o D. FINISCO QUANDO SCRIVO NELLA STRINGA "fine".

PROCEDIMENTO:

* STRUTTURA DATI: PER EFFETTUARE IL PROGRAMMA DEVO MEMORIZZARE TUTTI I NUMERI DELLA STRINGA? NON NECESSARIAMENTE DEVO MEMORIZZARLI IN UN VETTORE!

* LETTURA: sscanf(s, "%s ␣ %c ␣ %d", nome, &c, &p)

* CICLO: while(stringa != fine) IN MODO DA RICICLARE IL CICLO FINO ALL'INSERIMENTO DELLA PAROLA fine

NOTA = PER CONFRONTARE 2 STRINGHE UTILIZZO strcmp(): RISPONDE CON
VALORE POSITIVO → 1 STRINGA < 2 STRINGA
ZERO → 1 STRINGA = 2 STRINGA
VALORE NEGATIVO → 1 STRINGA > 2 STRINGA

LEZIONE 26 - MATRICI

INTRODUZIONE [VEDI DISPENSE]

LEGGERE DA TASTIERA UNA MATRICE [CICLI ANNIDIATI]:

```
for(i=0; i<R; i++) {
    for(J=0; J<C; J++) {
        Printf("matrice = %d %d", i, J);
        Scanf("%d", &m[i][J]);}}
```

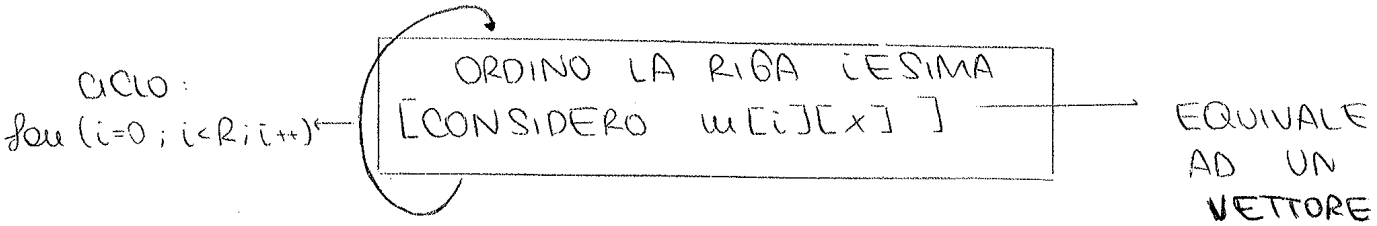
* **ESERCIZIO 1** - HO UNA MATRICE DI NUMERI REALI m[4][5]. VOGLIO FARE LA SOMMA DI TUTTI GLI ELEMENTI.
* DEVO PRENDERE L'ELEMENTO GENERICO m[i][J] A UN VALORE CONTENENTE LA SOMMA DEGLI ELEMENTI PRECEDENTI. QUESTO PER TUTTI GLI ELEMENTI DELLA MATRICE.

```
for(i=0; i<R; i++) {
    for(J=0; J<C; J++){ somm = somm + m[i][J];}}
```

[SE DOVESSI OPERARE SU UNA SOLA RIGA / COLONNA LE CONSIDERO COME VETTORI].

* **ESERCIZIO 2**: LEGGERE DA TASTIERA UNA MATRICE E CALCOLARE QUANTI SONO GLI ELEMENTI PARI PER CIASCUNA RIGA E QUANTI DISPARI PER CIASCUNA COLONNA.

ALGORITMO: ORDINARE LE RIGHE



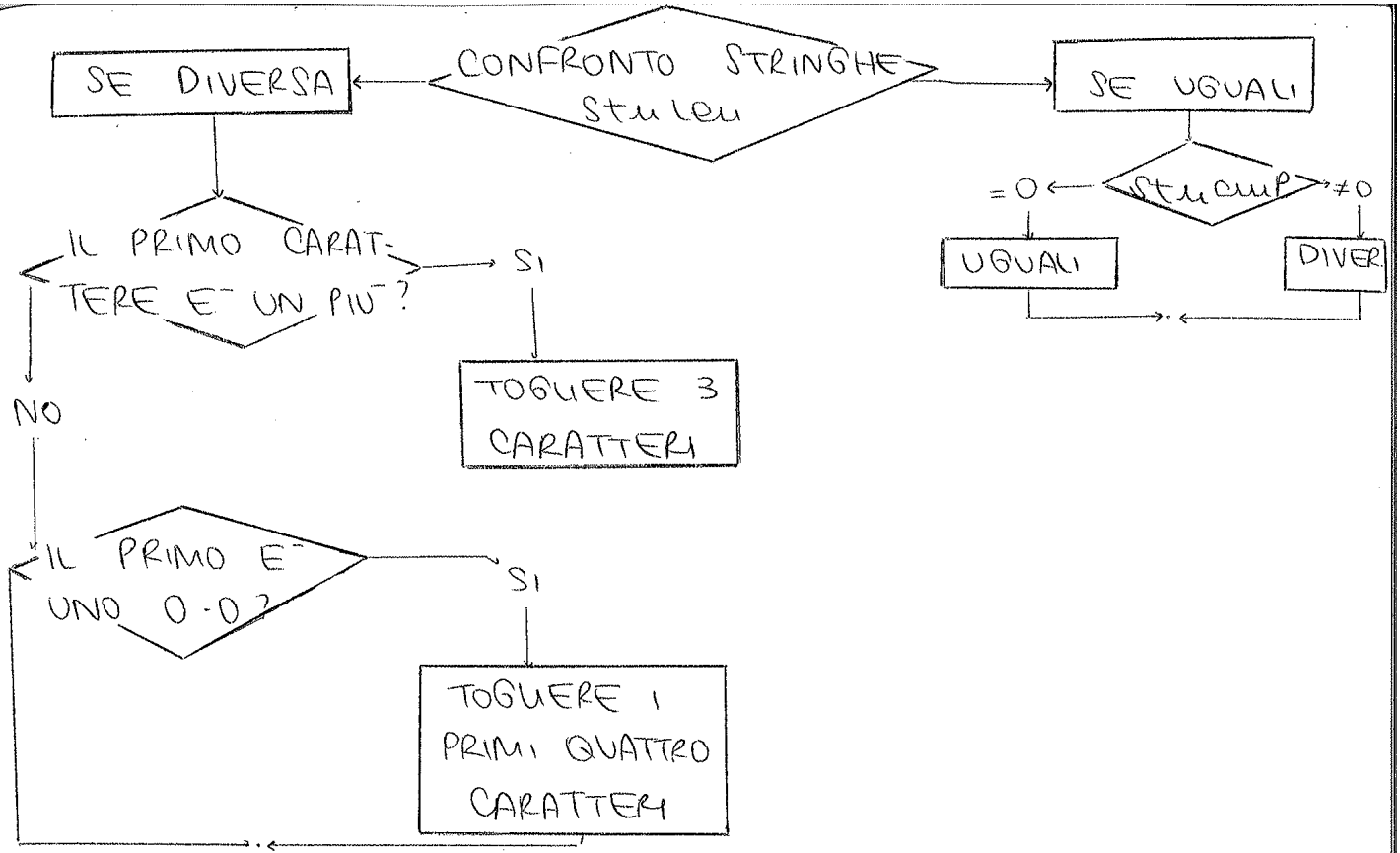
NOTA = CON LE MATRICI COME MINIMO 2 CICLI [for]: UN CICLO CHE OPERA SULLA RIGA COME SE FOSSE UN VETTORE E UN CICLO CHE RIPETA L'OPERAZIONE PER TUTTE LE RIGHE.

```c
#include <stdio.h>
#include <stdlib.h>
#define R 3
#define C 4
int main()
{   int m[R][C]={{3,-9,8,6},{1,6,7,3},{-3,8,-5,9}};
    int i,t,lung,q,j;
/*RIGHE*/
/*con questo for passo da una riga ad un'altra riga.
Opera, in pratica, con la matrice*/
    for(i=0;i<R;i++){
    lung=C;
/*con questo for considero vettori via via più piccoli*/
        for(j=0;j<C-1;j++){
/*con questo for più interno faccio eseguire
l'elemento minore*/
                for(q=0;q<lung-1;q++){
                if(m[i][q]<m[i][q+1]){
                    t=m[i][q];
                    m[i][q]=m[i][q+1];
                    m[i][q+1]=t;
                }
                }
        lung--;
        }
    }
/*utilizzo un altro for per stampare la matrice
per righe*/
for(i=0;i<R;i++){
    for(j=0;j<C;j++){
        printf("%d  ",m[i][j]);
/*non serve l'a prima della matrice*/
/*ATTENZIONE alla spazio dopo %d*/
    }
    printf("\n");

}
```

```c
printf("\n++++++++++++ \n\n");
/*COLONNE*/
    for(i=0;i<C;i++){
    lung=R;
        for(j=0;j<R-1;j++){
                for(q=0;q<lung-1;q++){
                if(m[q][i]<m[q+1][i]){
                    t=m[q][i];
                    m[q][i]=m[q+1][i];
                    m[q+1][i]=t;
                }
                }
        lung--;
        }
    }
for(i=0;i<R;i++){
    for(j=0;j<C;j++){
        printf("%d  ",m[i][j]);
    }
    printf("\n");
```

CONFRONTARE DUE RIGHE EQUIVALE A CONFRONTARE DUE VETTORI.
* IL CONFRONTO DEGLI ELEMENTI DI DUE VETTORI RICHIEDE, ANCH'ESSO, UN CICLO. `for(k=0; k<N; k++) {`
`if(v1[k] != v2[k]) { flag=0; }}`
CON flag INIZIALIZZATO A 1; MI AIUTA A VEDERE SE SONO UGUALI [flag=1] O SONO DIVERSI [flag=0]
v1[k] ↔ ri = m[i][k] rj = m[j][k] ↔ v2[k]
i E j INDICANO LA RIGA - k INDICA L'ELEMENTO DELLA RIGA.

```c
#include <stdio.h>
#include <stdlib.h>
#define R 4
#define C 4
int main()
{   int m[R][C]={{0,0,0,0},{1,2,3,4},{0,0,0,0},{5,7,8,5}};
    int i,j,k,flag;
    for(i=0;i<R;i++){
        for(j=i+1;j<R;j++){
            flag=1;
            for(k=0;k<C;k++){
                if(m[i][k]!=m[j][k]){
                    flag=0;
                }
            }
            if(flag==1){
                printf("Righe %d %d uguali",i,j);
            }
        }
    }
    return 0;
}
```



NOTA : POTREI OPERARE SULLA MATRICE ELIMINANDO IL PREFISSO INTERNAZIONALE

L'ESERCIZIO POTEVA ANCHE ESSERE SVILUPPATO COSÌ:

```
for(i=0; i<10; i++) {
    gets(s)
    sscanf("%s %d", citta', &h);
```

```
char s[90+1]
char citta'[20+1]
int h
```

```
[/* NON LI LEGGO COME VETTORI MA
 COME VARIABILI */]
    if (h > max) { max = h;
/* PER COPIARE UNA STRINGA IN UN'ALTRA
 STRINGA */
strcpy (citta'max, citta'); }
```

INIZIALIZZO:
max = 0;

```
char citta'max
[20+1]
```

[NOTA: COME INIZIALIZZO UNA STRINGA?
COME CONSIDERO UNA STRINGA VUOTA?
S STRINGA VUOTA È RAPPRESENTATA
DA UN CARATTERE, QUELLO DI FINE
STRINGA]

```
  somma = somma + h; }
```

INIZIALIZZO:
somma = 0;

```
printf("citta': %s, h max = %d", citta'max, max);
printf("media = %f", (float) somma/10);
```

*ESERCIZIO 2: SUPPONGO DI LEGGERE DA TASTIERA UNA SEQUENZA DI STRINGHE DEL TIPO: <nome squadra> <Punti> TERMINO CON L'INVIO. NON È QUINDI DEFINITO A PRIORI IL NUMERO DI ELEMENTI CHE INTRODUCO. VOGLIO ORDINARE LE SQUADRE IN BASE AI PUNTI / POI IN ORDINE ALFABETICO

DATI: IN QUESTO CASO [NON COME NELL'ES. 1] DEVO NEC. MEMORIZZARE LE INFORMAZIONI IN DUE MATRICI: UNA CONTENENTE IL NOME DELLE SQUADRE [MATRICE DI CARATTERI] L'ALTRA CONTENENTE I PUNTEGGI [MATRICE DI INTERI] DEVO, INOLTRE, DIMENSIONARE LA MATRICE AD UN NUMERO R DI RIGHE ANCHE SE NON LA "RIEMPO" TUTTA

POPOLARE MATRICE: char m[R][C] E IL VETTORE int P[R]

do while

LEGGO LA STRINGA → gets

LA SCORPORO [SEPARO NOMI - PUNTI] → sscanf

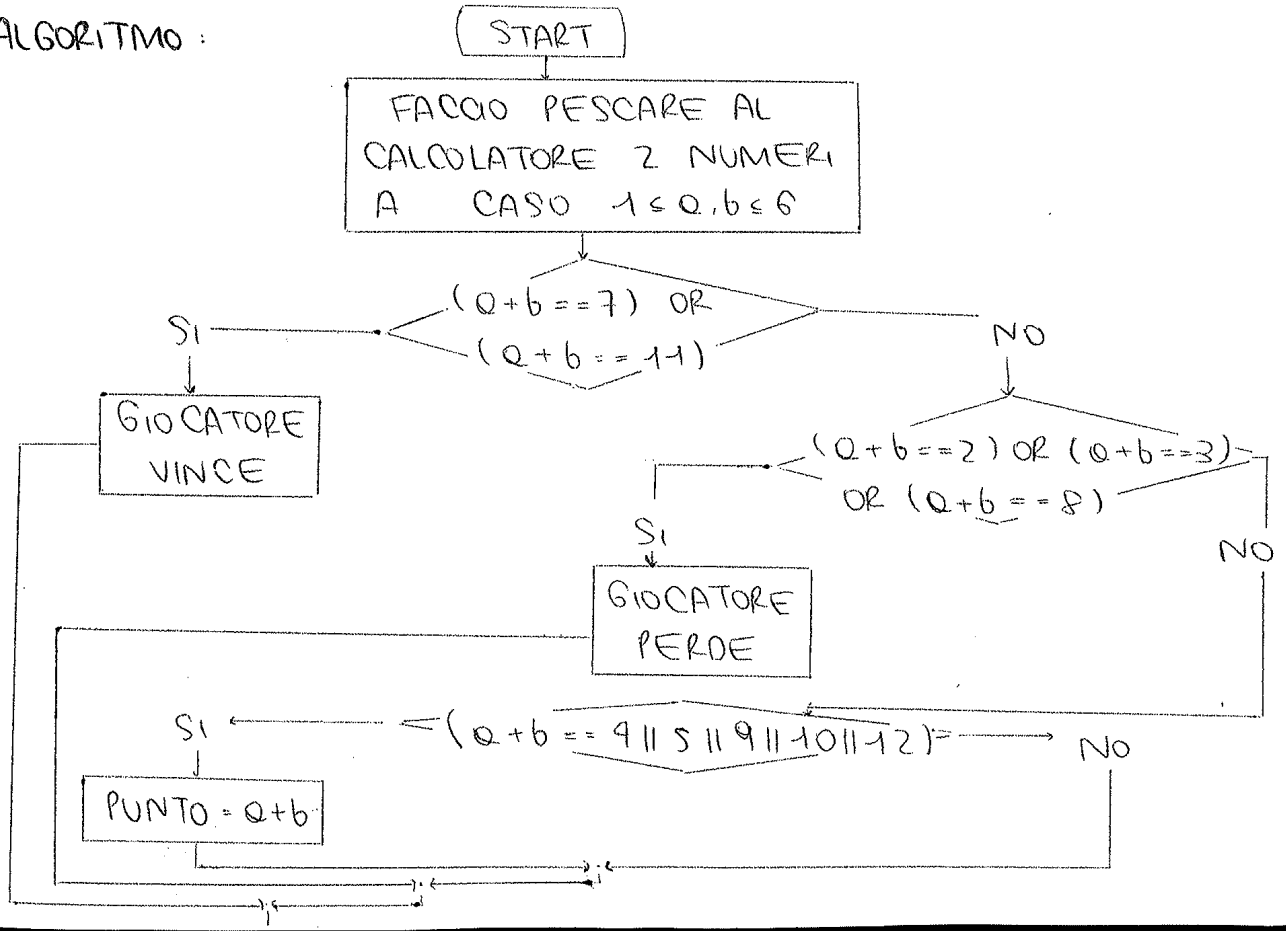
ESERCIZIO - GIOCO DEI DADI

UN GIOCATORE LANCIA DEI DADI. OGNI DADO HA SEI FACCE. QUESTE FACCE CONTENGONO 1, 2, 3, 4, 5, 6 PUNTINI. DOPO CHE I DADI SI SONO FERMATI, SI CALCOLERA' LA SOMMA DEI PUNTINI SULLE FACCE RIVOLTE VERSO L'ALTO.

AL PRIMO GIRO:
- SE LA SOMMA E' 7 O 11 => IL GIOCATORE VINCE
- SE LA SOMMA E' 2 O 3 O 8 => IL GIOCATORE PERDE
- SE LA SOMMA E' 4, 5, 6, 12, 9 O 10 => TALE SOMMA DIVENTA IL PUNTEGGIO [PUNTO] DEL GIOCATORE.

AL SECONDO GIRO [CHE INIZIA SOLO SE IL GIOCATORE NON HA GIA' VINTO O PERSO] E NEI GIRI SUCCESSIVI:
- SE IL GIOCATORE OTTIENE NUOVAMENTE PRIMA IL PROPRIO PUNTEGGIO E POI UN 7 => VINCE
- SE IL GIOCATORE OTTIENE PRIMA UN 7 [PRIMA DI RIOTTENERE IL PROPRIO PUNTEGGIO => PERDE
- SE IL GIOCATORE OTTIENE 11 => VINCE
- SE IL GIOCATORE OTTIENE 2, 3, O 8 => PERDE
- SE DOPO 10 GIRI NON VINCE => PERDE

ALGORITMO:



```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main()
{   int a,b,punto=0,c,d;
    int flag=0,cont=0,flag1=0,i=0;
    printf("Primo giro:");
    srand(time(NULL));
    a=(1+rand()%6);
    b=(1+rand()%6);
    printf("\nDADO 1 = %d",a);
    printf("\nDADO 2 = %d",b);
    if((a+b==7)||(a+b==11)){
        printf("\nVince il Giocatore");
        flag1=1;}
    if((a+b==2)||(a+b==3)||(a+b==8)){
        printf("\nIl Giocatore perde");
        flag1=1;}
    else {punto=a+b;}
     if(flag1!=1){
     printf("\nGiri successivi:");
      while((cont==0)&&(i<10)){
      srand(time(NULL));
      c=(rand()%6+1);
      d=(rand()%6+1);
      printf("\nDADO 1 = %d",c);
      printf("\nDADO 2 = %d",d);
      if(c+d==punto)
        {flag++;};
      if((flag!=0)&&(c+d==7)){
        printf("\nIl Giocatore vince");
        cont=1;}
      if((flag==0)&&(c+d==7)){
        printf("\nIl Giocatore perde");
        cont=1;}
      if(c+d==11){
        printf("\nIl Giocatore vince");
        cont=1;}
      if((c+d==2)||(c+d==3)||(c+d==8)){
        printf("\nIl Giocatore perde");
        cont=1;}
        i++;}
     if(cont==0){
     printf("\nIl Giocatore perde");}}
return 0;
}
```

ESEMPIO COMPLETO:

```
#include <stdio.h>
int ff (int a, int *p);  RITORNA DEL VALORE DI a IL QUADRATO E IL CUBO
main() { int x, qa, ca;
         x = 2;
         qa = ff(x, &ca);
         Printf("quadrato = %d - cubo = %d ", qa, ca);
    return 0; }
int ff (int a, int *p) { int q;
                         q = a*a;
                        *p = a*a*a;
                         return q; }
```

* ESERCIZIO 1 - CALCOLARE IL MINIMO E MASSIMO DI UN VETTORE MEDIANTE UNA FUNZIONE

```
int main() {
    int v[N] = {1, 2, 3, ...};
    int max, min; ...
  ff(&min, &max, v, N)
```

int ff{v}

PER RITORNARE HO 2 POSSIBILITA':

- int ff:
  - MIN => return
  - MAX => Puntatore
- void ff:
  - MIN => Puntatore
  - MAX => return

SCEGLIENDO IL 2° MODO:

```
void ff(int *Pmin, int *Pmax , int v, int N)
  void ff(int *Pmin, int *Pmax , int v, int N){
  max = v[0];     for(i=0; i<N; i++){
                    if(v[i] > max) {
                      max = v[i]; }}
  *Pmax = max; }
```

BLOCCO DEL MAX

[BLOCCO DEL MIN]

* **ESERCIZIO PROPOSTO** DA FARE A CASA = SIA DATO UN VETTORE E SIANO LETTI DA TASTIERA 2 NUMERI [S1 E S2]. QUANTI ELEMENTI DEL VETTORE SONO < S1 E QUANTI SONO > S2 → DEVO ADOPERARE UNA FUNZIONE!

* **ESERCIZIO PROPOSTO** = SIA DATA UNA MATRICE. SI CALCOLI, MEDIANTE UNA FUNZIONE, QUANTE SONO LE RIGHE = 0 E QUANTE SONO LE COLONNE UGUALI A = 0

LEZIONE 32 - SVILUPPO DI ESERCIZI

* ESERCIZIO 1 - LEGGERE DA TASTIERA UNA SEQUENZA DI NUMERI E SI VALUTI QUANTI DI QUESTI SONO COMPRESI TRA DUE VALORI S1 E S2, LETTI DA LINEA DI COMANDO. LA FINE E' QUANDO INTRODUCO IL NUMERO ZERO.

LINEA DI COMANDO: PIPPO (NOME DEL PROGRAMMA) S1 S2 (CIFRE)

PER ACCEDERE ALLA LINEA DI COMANDO: int main(int argc, char *argv[])

SCRITTO COME VETTORE DI PUNTATORI / STRINGHE = MATRICE DI CARATTERI

PER USCIRE DAL PROGRAMMA: · return 1 NUMERO ≠ DA ZERO
· exit( )

NOTA: SE SONO NELLA FUNZIONE IL return MI FA TORNARE AL PROGRAMMA, NON MI FA USCIRE. IN CASO DI FUNZIONE USO exit

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   /* IMPORTANTE 1 - 11 */
4   int main(int argc, char *argv[])
5   { int s1,s2,n,flag=0;
6       if(argc!=3){
7           printf("ERRORE");
8           return -1;}
9     sscanf(argv[1],"%d",&s1);
10    /* s1=atoi(argv[1]); */
11    sscanf(argv[2],"%d",&s2);
12      do{
13          printf("Numero = ");
14          scanf("%d",&n);
15          if(n!=0){
16           if((n>s1)&&(n<s2)){
17           flag++;}}}
18      while(n!=0);
19    printf("Risultato = %d",flag);
20  return 0;}
```

LEZIONE 33 - TIPI DI AGGREGATI

INTRODUZIONE: TIPI AGGREGATI = INSIEME DI OGGETTI DI DIVERSO TIPO [VEDI DISPENSE]

* **DICHIARAZIONE:**

NOTA = IN C VENGONO CHIAMATI "STRUTTURE"

PER INTRODURRE IL NUOVO OGGETTO GLI DO' UN NOME. PER DICHIARARE TALE OGGETTO, CHE E' UNA STRUTTURA, DEVO INDICARE CHE CONTIENE ELEMENTI DI DIVERSO TIPO

ESEMPIO: VOGLIO TRATTARE UNA STRUTTURA - L'OGGETTO LIBRO.

```
Struct LIBRO {
    [scatole che trovo nello scatolone]
    · char titolo [80];
    · char P [9];
    · float PREZZO; }
```

← DENTRO IL LIBRO CREATI UNA SCATOLA DI NOME TITOLO, CARATTERIZZATA DALL'ESSERE UN VETTORE DI CARATTERI

ALTRO → ALTRO TIPO DI ELEMENTI [FLOATING] SEMPRE CONTENUTI NELLO SCATOLONE LIBRO

STO DICENDO AL PROGRAMMA CHE OLTRE A TRATTARE CON INTERI, CARATTERI ECC., TRATTA CON LA STRUTTURA "LIBRO" CONTENENTE CIO' CHE HO INDICATO

* INOLTRE, AL CALCOLATORE FORNISCO UN'ALTRA **INDICAZIONE**: CON GLI ELEMENTI E LE INFORMAZIONI CONTENUTI NELLA STRUTTURA [SCATOLONE] CREAMI UN NUMERO PREDEFINITO DI SCATOLE ALLE QUALI DO DEI NOMI:

```
Struct LIBRO L1, L2;
```

* SE VOGLIO FARE RIFERIMENTO AD UN CAMPO IN PARTICOLARE

<nome variabile>.<nome campo>

COSI' POSSO ESPRIMERE NON TUTTO IL CONTENUTO DELLA STRUTTURA MA LE **SINGOLE COMPONENTI** ALL'INTERNO DI ESSA.

LEZIONE 34 - ESERCIZI SU STRUTTURE

* **ESERCIZIO 1** - LEGGERE DA TASTIERA DELLE COPPIE DEL TIPO: nome, peso E CALCOLARE QUALE COPPIA HA IL PESO MAGGIORE. SI ADOPERI UNA STRUTTURA

& COME RAPPRESENTO I MIEI **DATI** - DEVO DEFINIRE LA STRUTTURA CHE DOVRA CONTENERE L'OGGETTO "COPPIA" E I DUE CAMPI: nome E peso. PER DEFINIRE QUESTO TIPO DI STRUTTURA = Struct COPPIA {

```
char n[80];   ← nome = VETTORE DI CAR.
int peso; }
```

QUESTA E' LA DEFINIZIONE DEL NUOVO TIPO "COPPIA"

& AVENDO QUESTO TIPO DI DATI, NON E' NECESSARIO MEMORIZZARE TUTTE LE INFORMAZIONI. NON HO, QUINDI, BISOGNO DI STRUTTURE DATI COMPLESSE, MA MI BASTA SALVARE I DATI IN DUE VARIABILI - LA VARIABILE DI <u>COPPIA</u> E QUELLA DEL <u>MAX</u>

```
Struct COPPIA P, Pmax
```

NOTA: VOGLIO LEGGERE 4 COPPIE

& PER LEGGERE DA TASTIERA:

METODO 1 = PRIMA LEGGO IL NOME → Printf / Scanf (%s, P.n)

DEVO METTERE L'INFO NELLA PARTE CHE FA RIFER. AL NOME

POI LEGGO IL PESO → Printf / Scanf (%d, &P.Pe)

METODO 2 - LEGGO LA STRINGA CON gets() E POI LA SCORPORO NEI CAMPI CHE LA COSTITUISCONO → sscanf

* **ESERCIZIO 2** - SULLE MATRICI QUANDO VENGONO RICHIAMATE NELLE FUNZIONI: **PARAGONE TRA VETTORI E MATRICI**

VETTORI	MATRICI
• funzione (int v[], int NN) ... funzione (V, L_NUM. ELEMENTI)	• funz (int m[ ][C], int RIGHE) RIGHE COLONNE ... funz (m, RIGHE)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{ int i; char s[80];
/* Qui ho definito la struttura*/
struct coppia {
  char n[30];
  int pe;};
/* Qui ho definito le variabili*/
struct coppia p,pm;
/* Legge la stringa e la suddivide in
campo "nome" e campo "peso" */
pm.pe=0;
for(i=0;i<4;i++){
    gets(s);
    sscanf(s, "%s %d", p.n,&p.pe);
/* Cerca il massimo */
if(p.pe>pm.pe){
    pm.pe=p.pe;}
    strcpy(pm.n,p.n);}
printf("max %s  %d",pm.n,pm.pe);
return 0;}
```

```c
#include <stdio.h>
#include <stdlib.h>
#define R 3
#define C 80
int ff(char m[][C], int RR, int CC)
/* cc è la colonna che deve essere considerata */
int main()
{ char m[R][C]; int i,min,val=0;
/* Ho letto e normalato la stringa*/
  for(i=0;i<R;i++){
    printf("STRINGA = ");
    scanf("%s", m[i]);}

/* Ora calcolo la stringa con numero di colonne minimo */
/* Inizializzazione minimo != inizializzazione massimo,
il minimo deve inizializzarlo al valore massimo */

min=80;
  for(i=0;i<R;i++){
/* ragiono sulla lunghezza len delle stringhe */
    if(strlen(m[i])<min){
      min=strlen(m[i]);}}
  for(i=0;i<min;i++){
    val=ff(m,R,i);}
printf("COLONNA %d NUMERO CIFRE &d",i,val);
return 0;}

int ff(char m[][C], int RR, int CC){
int j,cont=0;
for(j=0;j<RR;j++){
    if(isdigit(m[j][CC])==0){
        cont++;}}}
```

* ESERCIZIO 1 - LEGGERE DA TASTIERA DELLE COPPIE, COSTITUITE DA UN NOME E UN INTERO E SCRIVERLE SU UN FILE DI NOME PIPPO.txt

ALGORITMO: main

CODICE:

```
FILE * fp;
char NOME 80; int peso;
```

[DEVO APRIRE IL FILE]

```
fp = fopen("pippo.txt", "w");
```

IL FILE LO DEVO SCRIVERE NON LEGGERE

```
if(fp == NULL) {        ← SE L'OPERAZIONE FINISCE MALE
printf("ERRORE!");
return -1; }
```

```
printf("NOME = ");          printf("Peso = ");
scanf("%s", NOME);          scanf("%d", &peso);
```

↑ HO GLI ELEMENTI PER SCRIVERE UN CAMPO SUL FILE

```
fprintf(fp, "%s %d\n", NOME, peso);
```

CICLO do

FINITO IL CICLO POSSO CHIUDERE IL FILE

```
fclose
```

* ESERCIZIO 2 - PRENDO UN FILE DI TESTO E RISTAMPARLO IN RIGHE CENTRATE. IL FILE RICEVE COME ARGOMENTO SULLA RIGA DI COMANDO:

- IL NOME DEL FILE D'INGRESSO
- IL NOME DEL FILE D'USCITA
- LA LUNGHEZZA MASSIMA (LM) DI UNA RIGA NEL FILE DI USCITA

IL PROGRAMMA LEGGA IL FILE IN INGRESSO E LO COPI SUL FILE DI USCITA CENTRANDO IL TESTO DI OGNI SINGOLA RIGA RISPETTO ALLA LUNGHEZZA MASSIMA.

IN PRATICA, LE RIGHE PIU' LUNGHE VANNO TRONCATE ALLA LUNGHEZZA LM E LA RIMANENTE PARTE VA CONSIDERATA COME UNA NUOVA RIGA. INVECE LE RIGHE PIU' CORTE DI LM DEVONO ESSERE CENTRATE AGGIUNGENDO UN' OPPORTUNA SEQUENZA INIZIALE DI SPAZI

SOLUZIONE:

* CONTROLLO GLI ARGOMENTI
* APRO I FILE
* LEGGO E CONTROLLO LA LUNGHEZZA MASSIMA DELLA RIGA
* LEGGO IL FILE DI INPUT UNA RIGA PER VOLTA E PER OGNI RIGA:
  - SE LA SUA LUNGHEZZA < LM => STAMPO LA RIGA PRECEDUTA DA UN NUMERO DI SPAZI PARI A (LM - RIGA)/2
  - ALTRIMENTI SE LA LUNGHEZZA > LM => NE STAMPO I PRIMI LM CARATTERI, AVANZO L'INDICE DI INIZIO RIGA DI LM CARATTERI, STAMPO CENTRATA LA RESTANTE PARTE

* ESERCIZIO 3 -

*In un file è memorizzata una matrice che contiene i voti conseguiti da 100 studenti in 30 esami diversi
* Gli studenti di cui non si conosce il numero massimo corrispondono alle righe mentre gli esami (numerati da 0 a 29)
* corrispondono alle colonne.
* Poichè la matrice è molto sparsa (ogni studente ha dato pochi esami) essa è memorizzata nel file indicando
* su ogni riga un singolo elemento: indici I e J seguiti dal valore dell'elemento in posizione (I,J), ovvero
* il voto conseguito dallo studente I-esimo nell'esame J-esimo.
* Il nome del file è passato come primo parametro sulla linea di comando.
* Si desidera calcolare e presentare in output la media dei voti di ciascun esame.
*
* SOLUZIONE:
* Non serve memorizzare la matrice dentro al programma !!!
* Occorre memorizzare per ogni esame:
* - Il numero di studenti che l'hanno sostenuto
* - la somma dei voti conseguiti da questi studenti.
* Occorre quindi un vettore di struct

*	n_studenti	totale_voti
* esame[0]	0	0
* esame[1]	2	60
* esame[2]	5	100
* esame[...]	...	...

*
* Controllo dei parametri e apertura del file
* si leggono tutte le righe del file (while + fgets)
* per ogni riga
* - lettura di I,J,VAL
* - si incrementa il numero di studenti dell'esame J e si somma VAL al suo totale dei voti
* - alla fine si calcola la media per ciascun esame */

LEZIONE 37 - ESERCIZI ESAME

TEMA D'ESAME = 3 ESERCIZI DI TEORIA VALGONO 2 PUNTI L'UNO

1 - SUPPONIAMO DI AVERE 2 NUMERI IN COMPLEMENTO A DUE CHE SONO CODIFICATI IN ESADECIMALE E CHE VALGONO:

PRIMO NUM. → FC 8 9 ← SECONDO NUM

- VOGLIO L'EQUIVALENTE IN DECIMALE
- VOGLIO FARE LA SOMMA E VEDERE SE C'E' OVERFLOW

SVOLGIMENTO

F C

1111 1100 ← QUESTI DEVO RICORDARLI: VEDI TEORIA

NUMERO IN COMPLEMENTO A 2:

- E' UN NUMERO NEGATIVO = $-n$ QUANTO VALE n?

$* = -2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 = \dots =$

* CALCOLO CON L'ALGORITMO SPECIFICO POSSO CALCOLARE $+n$ [VEDI TEORIA] - PRIMA INVERTO BIT A BIT POI SOMMO 1

00000011 + 1

INVERSIONE BIT A BIT { 1 = 0, 0 = 1

00000100 ⇒ +4 ⇒ IL NUM. DI PARTENZA = -4

8 9

1000 0100 - ESSENDO IL NUMERO DEGLI "1" MOLTO BASSO CONVIENE USARE IL PRIMO METODO

$= -2^7 + 2^2 = -128 + 4 = -124$

SOMMA: 11111100 + 10000100

10000000 $= -2^7 = -128$

84

-129 - 9 ← FC

SE I SEGNI DEGLI ADDENDI SONO DIVERSI = MAI OVERFLOW

2 - HO UNA FUNZIONE BOOLEANA $f_1(x,y,z) = x\bar{y}\bar{z} + \bar{x}yz$

$f_2(x,y,z) = x\bar{y}\bar{z} + \bar{x}\bar{y}z$

CAPIRE SE $f_1 = f_2$???

DEVO CALCOLARE E CONTRONTARE LE TABELLE DELLA VERITA' DELLA FUNZIONE

· LEGGERE IL FILE DA LINEA DI COMANDO:

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

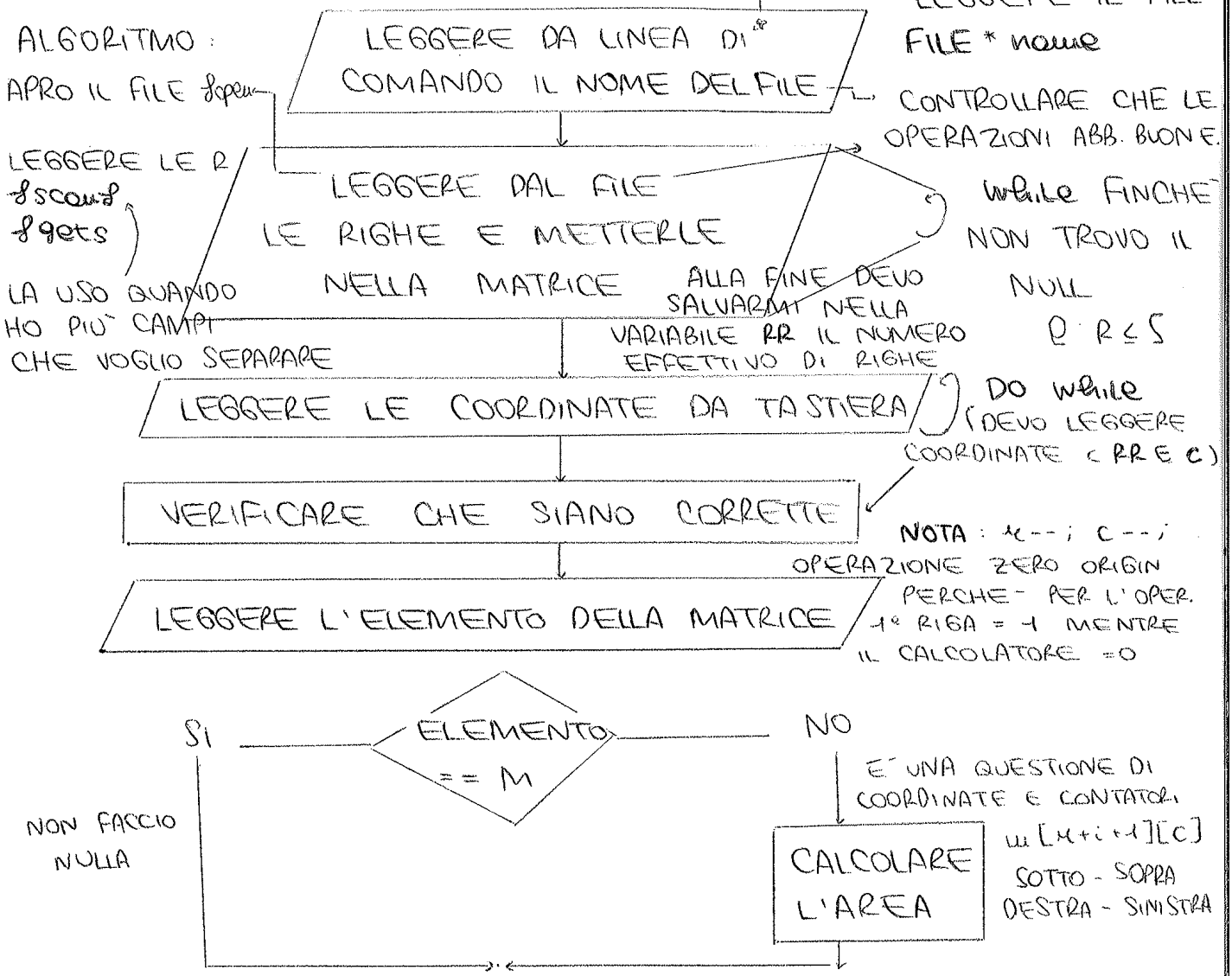
· DEFINISCO LA MATRICE:

char m[R][C+1];

argv[0] = nome
argv[1] = contenuto

LEGGERE IL FILE:
FILE * nome

ALGORITMO:



LEZIONE 38 - SIMULAZIONE ESAME

ALL' ESAME HO UN FOGLIO CON LE FUNZIONI

DOMANDA 1 -

DATO IL SEGUENTE NUMERO SU 6BIT: 1 1 0 1 0 1

SCRIVERE IL NUMERO :
- BINARIO PURO
- MODULO E SEGNO
- COMPLEMENTO A 2

DOMANDA 2 -

SIANO DATI I NUMERI IN BASE 10 $x1 = +253$

$x2 = -310$

SCRIVERLI IN COMPLEMENTO A 2 SU 9 BIT

CALCOLARE $x3 = x1 + x2$

OVERFLOW ?

DOMANDA 3 - SI SUPPONGA DI UTILIZZARE UN CALCOLATORE IN CUI I NUMERI INTERI SONO RAPPRESENTATI SU 32 BIT. QUAL E' IL NUMERO MINIMO DI **BYTE** OCCUPATO DA:

```
Struct { char NOME [20];
         char COGNOME [20];
         char MATRICOLA [8];
         int eta'; } studente;
studente registro [100];
```

PROGRAMMA IN C : "BATTAGLIA NAVALE"

RISOLUZIONE

LEGGERE COME SE FOSSE SCRITTO IN BP, MS, C

BINARIO PURO 1 1 0 1 0 1 : $2^5 + 2^4 + 2^2 + 2^0 = 53$

MODULO E SEGNO : $(-1)[2^4 + 2^2 + 2^0] = -21$

COMPLEMENTO A 2 : $-2^5 + 2^4 + 2^2 + 2^0 = -11$