NUMERO: 2222A

ANNO: 2017

A P P U N T I

STUDENTE: Trabace Maria

MATERIA: Informatica - Prof. Mezzalama

INFORMATICA → scienza che manipola l'informazione attraverso i DATI

CONCETTO ← CODIFICA

UNITÀ DI MISURA - MULTIPLI

CHILO	K	10^3
MEGA	M	10^6
GIGA	G	10^9
TERA	T	10^{12}
PETA	P	10^{15}
EXA	E	10^{18}

2^{10} 2^{20} 2^{30} 2^{40} 2^{50} 2^{60} (*)

LINGUAGGIO CALCOLATORE → Binary [0,1]

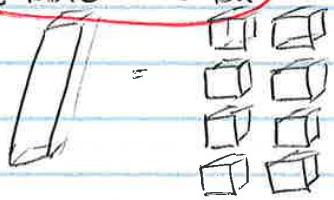
L'informazione viene gestita e memorizzata in INTERRUTTORI

"TRANSISTOR DIGITALE" ←

APERTO = 0

CHIUSO = 1

1 BYTE = 8 BIT



Normalmente:

1 numero intero = 2 BYTE

1 numero reale = 4 BYTE

3 MOMENTI:
- ELABORAZIONE → MIPS (milioni di istruzioni al secondo); MFLOPS (milioni di istruzioni aritmetiche su numeri reali al secondo)
- MEMORIZZAZIONE
- TRASPORTO

richiede un supporto fisico:
- RAM
- DISCO RIGIDO
- CD / DVD

↑↑ capacità, ↑↑ PC

ognuno con una propria capacità misurata in multipli di Byte

il calcolatore misura e ragiona in termini di potenze di 2 (*)

In particolare, in base a (*) $2^{10} = 1024 \approx 10^3$

Es 640 kB = 640 · 1024 B = 655360 Byte

1 GB = 1 · 1024 · 1024 · 1024 = 1073741567 Byte

→ "Bit al secondo" BPS /Es: Router a 2MB ⇒ 16MBPS

ELABORATORI → DI USO GENERALE (General Purpose) / DEDICATI (SPECIAL PURPOSE)

Possono anche avere la stessa CPU. Quelli "general" sono "dinamicamente programmabili", quelli "Specific" hanno un programma che è fisso, devono svolgere funzioni precise stabilite in fase di progetto.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 3 di 120

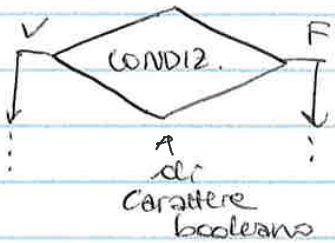
DIAGRAMMI DI FLUSSO e STRUTTURE DATI

acquisizione/rilascio dati

elaborazione dati (ad esempio, calcoli)

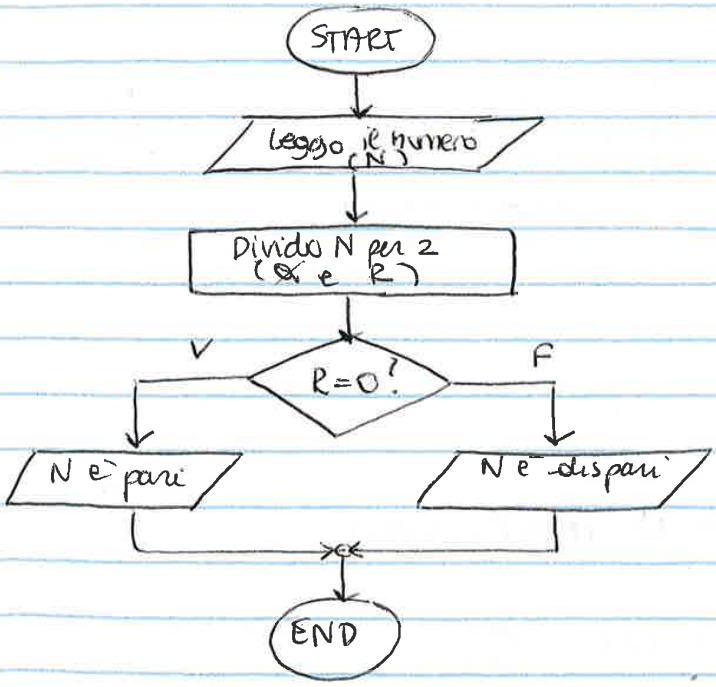
if - then - else

BLOCCO DI DECISIONE



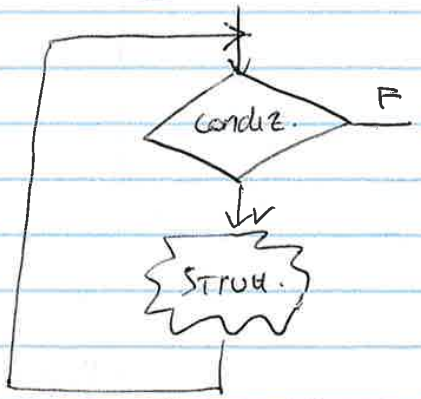
di carattere booleano

ES. Algoritmo che, dato un numero, stabilisca se è pari o dispari



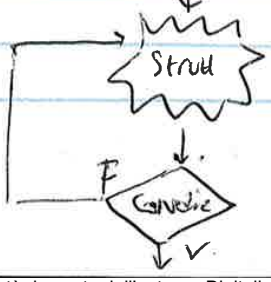
while - do

Gira sul vero

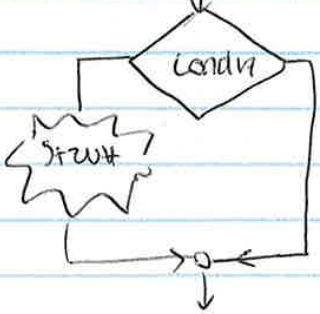


gira sul falso

do - while



if - then



© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 5 di 120

ESEMPI DI FLOWCHART

Es 1. Sia data una sequenza di 10 lettere rappresentante un frammento di DNA, si vuole verificare se in tale ~~sotto~~ sequenza è presente la sotto sequenza GA.

MASSIMA MEMORIA INTERNA (FISICAMENTE PRESENTE)

- La dimensione dell'ABUS determina il max num. di celle di memoria indirizzabili (fisico)
- La dimensione della DBUS indica la dimensione di una cella di memoria.

$$\text{MAX MEMORIA} = 2^{[\text{ABUS}]} \cdot [\text{DBUS}]\ [\text{bit}]$$

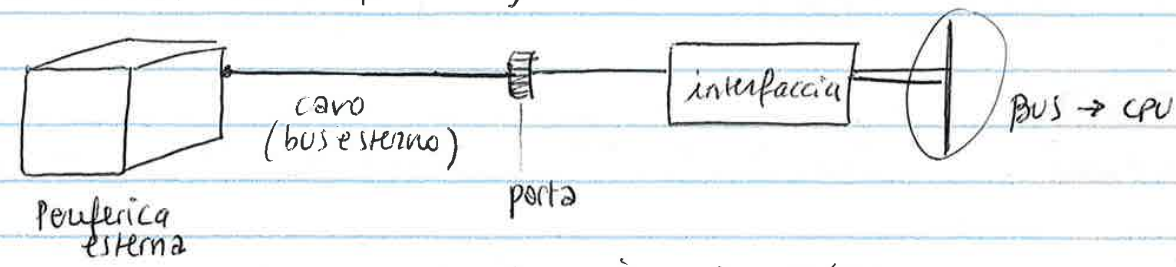
$2^{[\text{ABUS}]}$: numero di indirizzi di memoria

$[\text{DBUS}]\ [\text{bit}]$: quanto è "larga" una cella di memoria

Es: ABUS = 20 Bit, DBUS = 16 Bit

$$\text{MAX MEM} = 2^{20} \cdot 2\text{Byte} = 2\text{MByte}$$

In generale tutti i dispositivi periferici sono collegati da un'interfaccia (porta esterna, ad esempio USB):



Questo schema vale per tutte le unità di I/O

* MEMORIA → memorizza i dati e le istruzioni necessarie all'elaboratore x operare

- Indirizzamento (limitato dall'ABUS)
- Parallelismo
- Capacità, espressa in BYTE)
- Tempo di accesso (velocità con cui vado a leggere o a scrivere un dato in memoria)

- MEMORIA INTERNA → è allo stato solido (chip), veloce (~ns)
  - RAM → lettura/scrittura dal µP → volatile
  - ROM → solo lettura dal µP → non volatile
- MEMORIA ESTERNA → è molto lenta (~ms)
  - MAGNETICA → es. CD
  - OTTICA → DVD

SRAM "statica" + veloci, + costose

DRAM "dynamic"

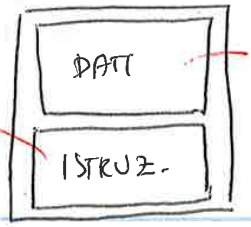
MEMORIA CENTRALE

SIST. OPERATIVO PROGRAMMI	RAM	~ 4 GB
MEMORIA VIDEO	RAM VIDEO	~ 1 GB
PROGRAMMA DI AVVIO	ROM	~ 128 KB

EROM ↔ PROM

LINGUAGGIO C

Programma:



parte dichiarativa (nome e tipo di variabili)

parte esecutiva

Es:

```
int main () {
    float x1, x2, a, diff, err;
    printf ("numero = ")
    scanf ("%f", &x1)
    {
    ;
    }
return 0;
}
```

dichiarativa

esecutiva

Es:

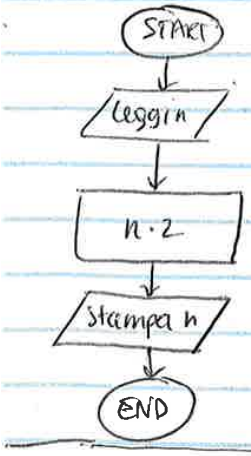
```
#include <stdio.h>
int main () {
    float a;
    a = 3.2;
    a = a + 0,5;
    printf ("%f", a);
return 0;
}
```

libreria

fattore di conversione

Es

Programma che acquisisce un numero da tastiera, lo moltiplica per 2 e stampa a video.




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

44.49

NB. x
Scambiare le variabili uso una variabile di appoggio:

```
int a, b, t;
:
t = a;
a = b
b = t;
:
```

overflow in un binary : il valore non è rappresentabile sullo stesso numero di bit

Se ho 3 bit il num. max rappresentabile è $111 = (7)_{10}$

LEZ 7. **SISTEMA ESADECIMALE** : $B = 16$ $\{0,1,2,3,4,\dots 9, A, B, C, \dots F\}$

Quando opero con un numero consistente di cifre è utile raggrupparle a 4 a 4

4 bit → $2^n = 16$ combinazioni → cifre $0 \div 15$

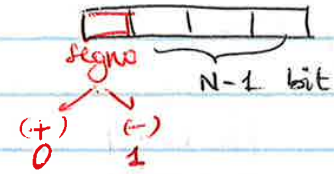
Es: $(1011\ 1001)_2$ $(1001\ 0001)_2$

$(B\ 9)_{16}$ $(9\ 1)_{16}$

$(1FA)_{16} = (0001\ 1111\ 1010)_2$

NUMERI RELATIVI SEGNO + VALORE

- **MODULO E SEGNO** $-(2^{N-1}-1) \le x \le 2^{N-1}-1$



$(+6)_{10} \Rightarrow (0110)_{M\&S}$

$(-6)_{10} = (1110)_{M\&S}$

$(-2)_{10}$ (su 4 bit) $\Rightarrow$ 1 0 1 0 $_{M\&S}$

Non viene molto adoperata perché non abbiamo + un sistema posizionale e le operazioni non sono + valide

$-(2^{N-1}) \le x \le (2^{N-1}-1)$

- **COMPLEMENTO A 2** . Tutti i bit hanno un peso, compresi i bit dei segni

→ la prima cifra è negativa : 0 1 1 0 1 0 1 0

$-2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

Es: $(101101)_{CA2} = -2^5 + 2^3 + 2^2 + 1 = -32 + 13 = (-19)_{10}$

Se la prima cifra è 0, allora il numero è sicuramente positivo ⇒ M&S ≡ CA2

CA2 : { 0110 → +6 ≡ M&S numero positivo
1011 → −8 + 2 + 1 = (−5)$_{10}$

Da decimale a CA2 ! Se num > 0 allora M&S ≡ CA2 ;
se num < 0 : (−4) • Parto dal numero opposto, quindi dal positivo : $(+4)_{10} \rightarrow (0100)_{M\&S}$

- inversione bit a bit ⇒ 1011
- Gli sommo 1 → 1011 +1

(Implementazione) per ottenere l'opposto.

$(1100)_{CA2} = (-4)_{10}$

Sottrazione

Es: $(+3) - (+1) \Rightarrow$ $0011 - 0001 = 0010$

oppure somma col Complemento

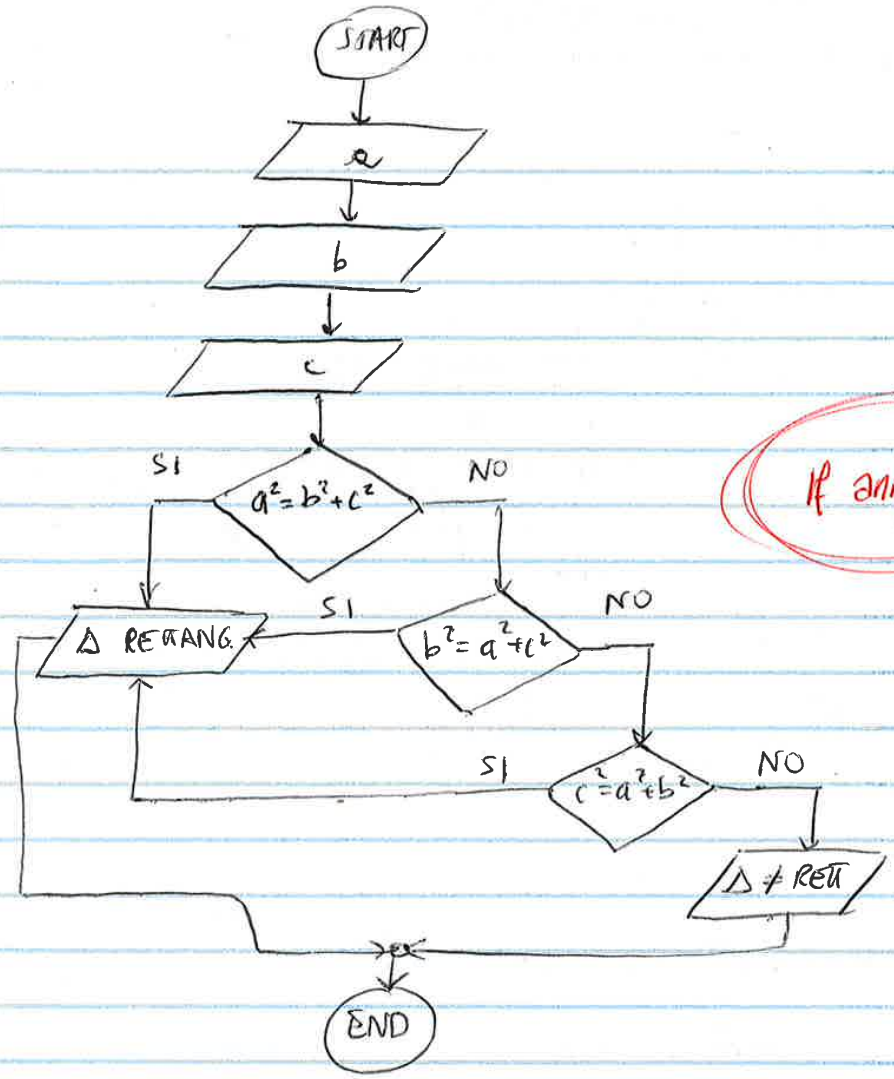
$0011 + 1111 = 0010$ (riporti 1 1)

OVERFLOW IN CA2

- operandi discordi → MAI
- operandi concordi → Se il risultato ha segno opposto

FLOW CHART

ok

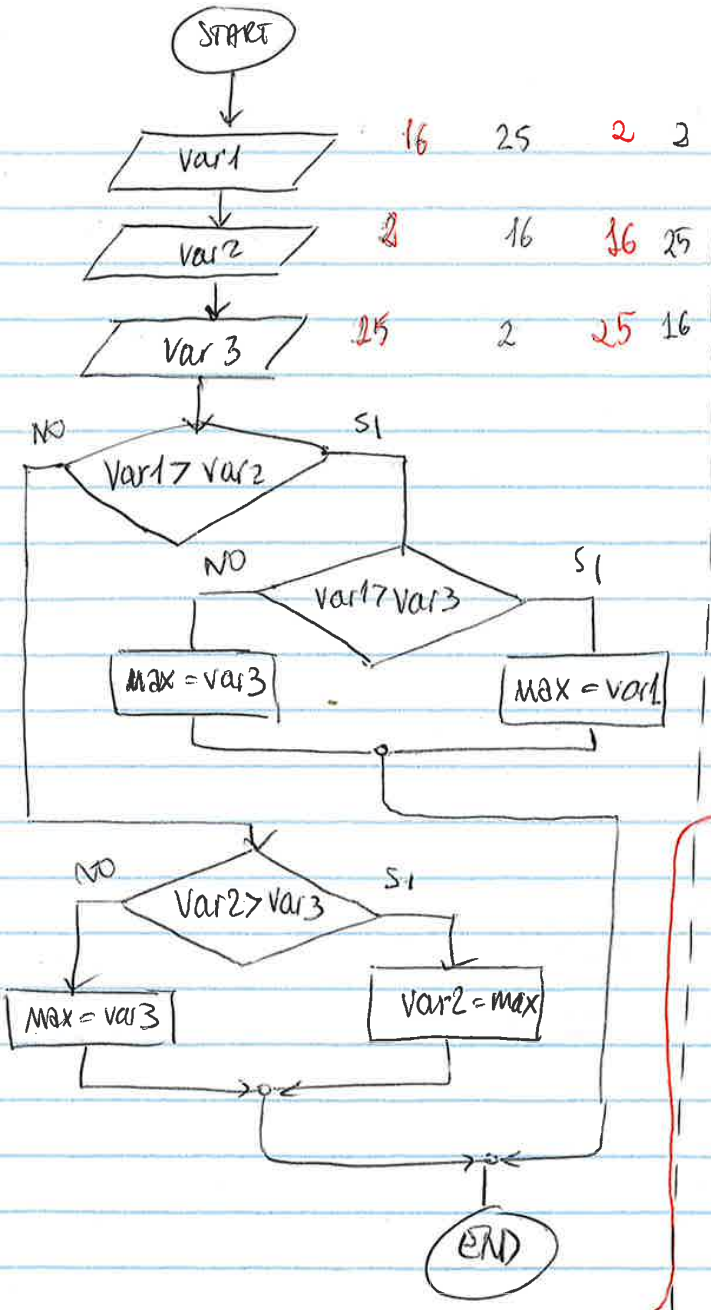


If annidati

C:

```
# include <stdio.h>
# include <stdlib.h>
int main () {
    int a, b, c;
    printf (" Inserisci i lati del triangolo : \n");
    scanf (" %d \n %d \n %d ", &a, &b, &c );
    if (( a*a = b*b + c*c) || ( b*b = a*a + c*c) || ( c*c = a*a + b*b))
        { printf (" TRIANGOLO RETTANGOLO. ");
        }
    else { printf ( "TRIANGOLO NON RETTANGOLO. ");
        }
    return 0;
}
```

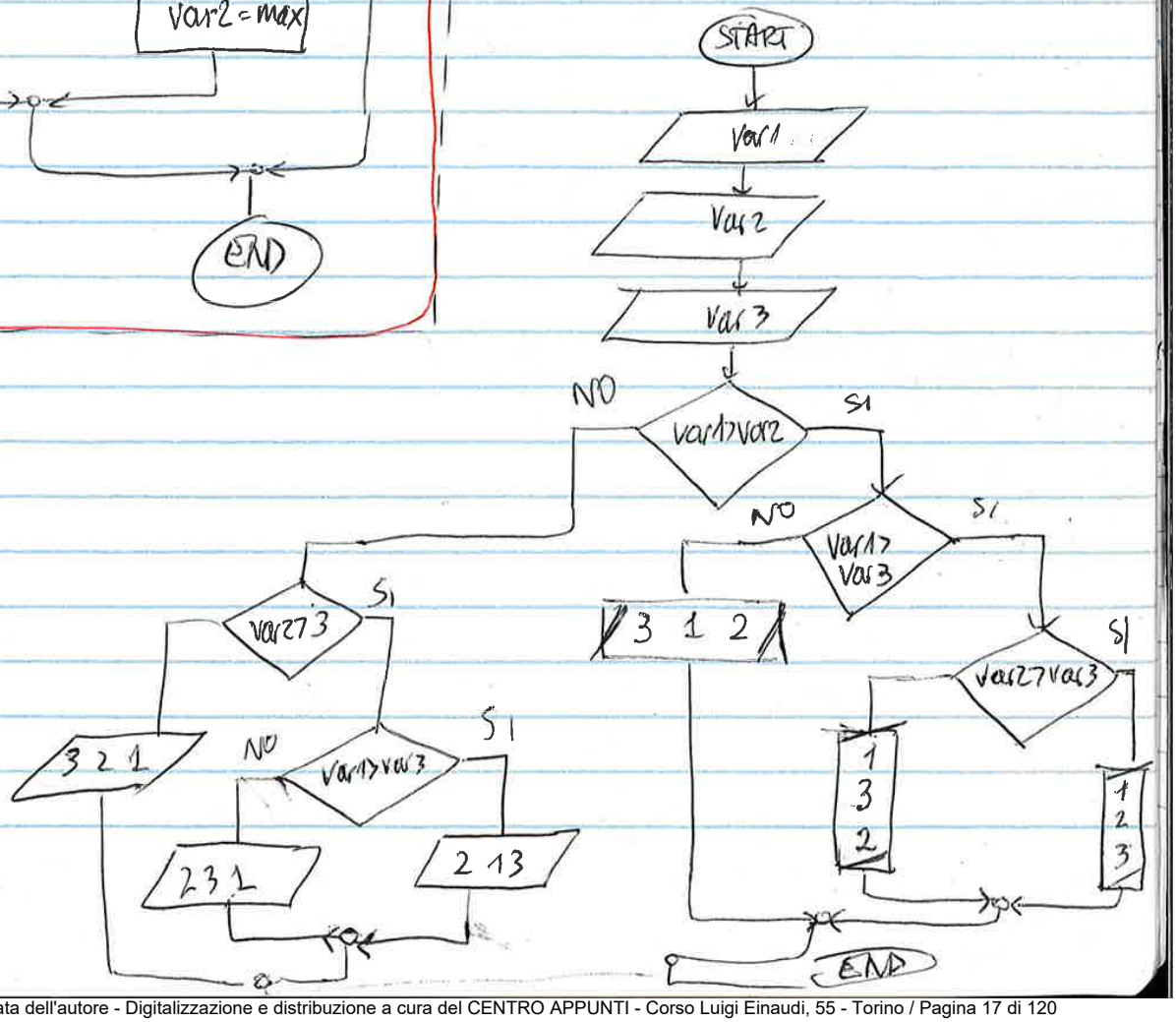
ES 3 Individuare il max fra 3 valori memorizzati in 3 variabili.



16 25 2 2
2 16 16 25
25 2 25 16

```
#include <stdio.h>
#include <stdlib.h>
int main () {
    int val1 = 45, val2 = 36, val3 = 148;
    int max;
    if ( val1 > val2) {
        if (var1 > var3)
            max = var1;
        else
            max = var3;
    }
    else { if (var2 > var3)
            max = var2;
        else
            max = var3;
    }
    printf ("Il massimo valore e' %d.", max);
    return 0;
}
```

ES 4 Stampare in ordine decrescente 3 valori memorizzati in 3 variabili var1, var2, var3.



LOGICA BOOLEANA

(VERO → 1, FALSO → 0)

(F_1, F_2) ← VARIABILI BOOLEANE + OPERATORI LOGICI { AND, OR, NOT }

F_1 (AND) F_2

È vera solo se F_1 e F_2 sono contemporaneamente vere, falsa in tutti gli altri casi

TABELLA di VERITA':

F_1	F_2	F_1 AND F_2
0	0	0 (F)
0	1	0 (F)
1	0	0 (F)
1	1	1 (V)

precedenza
1) NOT
2) AND
3) OR

F_1 (OR) F_2

È vera se almeno una fra F_1 e F_2 è vera, falsa se sono contemporaneamente false

Tabella di Verità':

F_1	F_2	F_1 OR F_2
0	0	0
0	1	1
1	0	1
1	1	1

(NOT) F_1

Vera quando F_1 è falsa

Tabella di Verità':

F_1	NOT F_1
0	1
1	0

Quindi accanto all'algebra normale esiste l'algebra di Boole, in cui le variabili possono assumere solo 2 valori (V o F) e si eseguono con gli operatori logici nel seguente ordine di precedenza:

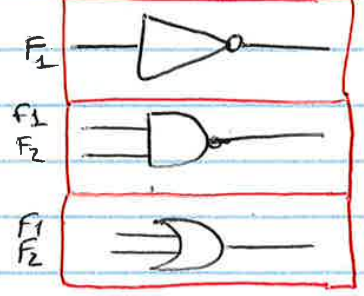
1) NOT
2) AND
3) OR

$f = x$ AND y OR z → equivale a $f = x * y + z$

(il NOT si indica con un apice a un tratto lungo)

- Gli operatori servono x fare circuiti: 1 bit → 1 var booleana

PORTE LOGICHE:

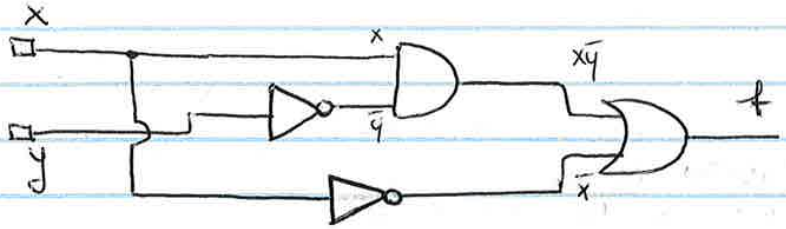


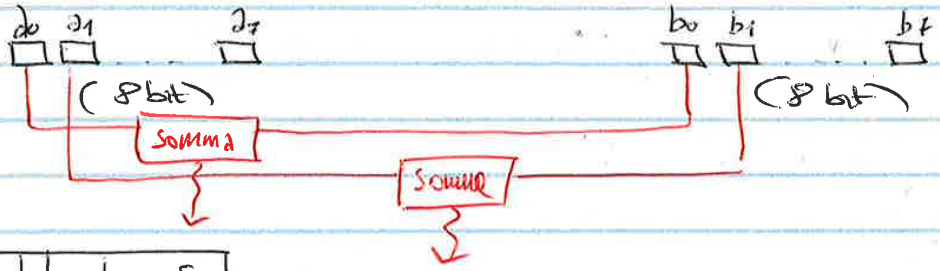
I bit in questo caso sono i valori della tensione

CIRCUITI Transistor + Porte Logiche (all'interno dei chip)

Come si passa dal problema al circuito?

$f = x\bar{y} + \bar{x}$, x, y variabili ("piedini" del processore)





a_0	b_0	S
0	0	0
0	1	1
1	0	1
1	1	0

→ f

Dalla tabella alla funzione: 2 regole:

- la funzione è data da tanti termini messi in "OR" fra di loro quanti sono le righe in cui la funzione vale 1:

$f = m_1 + m_2$

- ciascun termine è dato dal prodotto di tutte le variabili indipendenti ("AND")

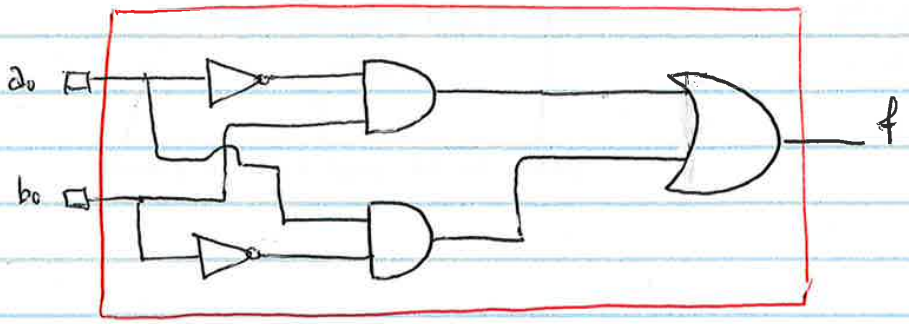
$m_1 = \bar{a}_0 b_0$ 1ª riga

$m_2 = a_0 \bar{b}_0$ 2ª riga

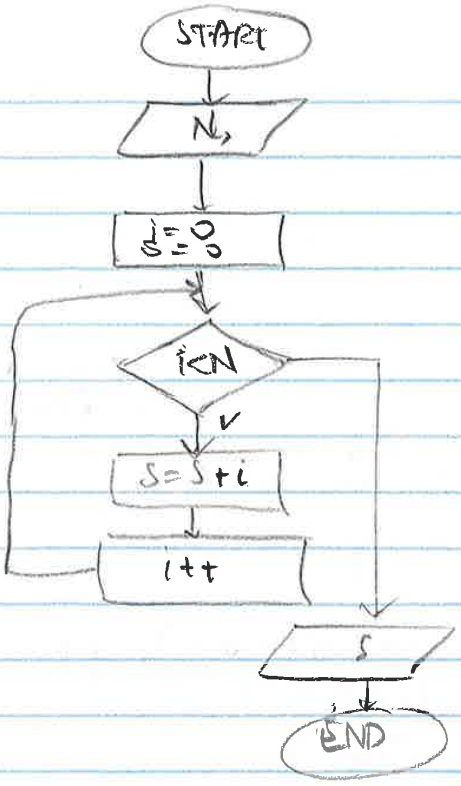
e la variabile è negata nelle stesse righe di prima laddove c'è uno zero.

$\Rightarrow$ $f = \bar{a}_0 b_0 + a_0 \bar{b}_0$

CIRCUITO (Sommatore di 2 bit) - ALU -



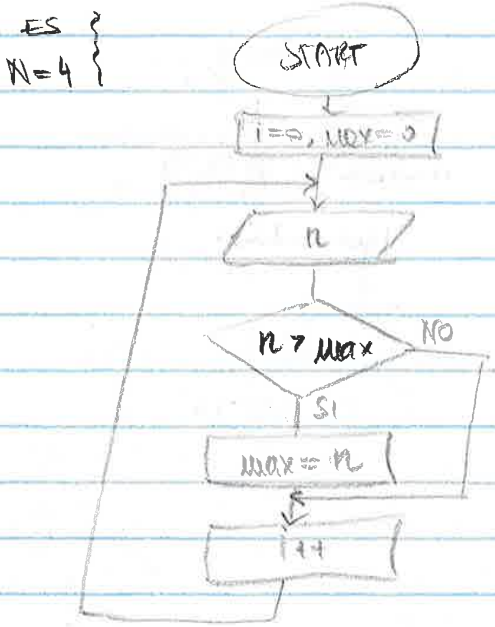
ES: Leggere un valore N, calcolare la somma dei primi N numeri interi e stamparla




```
#include <stdio.h>
#include <stdlib.h>
int main () {
    int N, S, i;
    S = 0;
    i = 0
    printf (" Inserisci un intero : \n ");
    scanf (" %d ", &N);
    while ( i<= N) {
        S = S+i;
        i++;
    }
    printf (" La somma dei primi %d numeri
        e %d. ", N, S );
    return 0;
}
```

ES Calcolare il max di N numeri introdotti da tastiera.

(soluzione)




```
#include <stdio.h>
#include <stdlib.h>
#define N 5
int main () {
    int i = 1;
    int max = -65000;
    int n;
    while ( i<=N) {
        printf ("Numero %d = ", i);
        scanf (" %d ", &n);
        if (n> max) {
            max = n;
        }
        i++;
    }
    printf ( " MAX = %d ", max);
    return 0;
}
```

ok

```
#include <stdio.h>
#include <stdlib.h>
#define N 5
int main ()
{
    int x, i=1;
    float sum=1, fatt=1;
    printf ("x = ");
    scanf ("%d", &x);
    while (i<=N) {
        fatt = fatt*i;
        sum = sum + (x^i)/fatt;
        i++;
    }
    printf ("exp = %f", sum);
    return 0;
}
```

1) Prog. che legge da tastiera delle temperature (int)
- Temp. Max
- Temp. Min
- Temp Min

- Quante temp sono comprese in un intervallo [n1, n2] (n1 e n2 da tastiera)

① N define

② Mi blocco se introduco T = -1000°C

2.1 2 numeri 1 byte (8 bit) in C2 → F3, 71
- valore decimale
- somma (overflow?)

2.2 $f_1 = xy\bar{z} + z\bar{y}$ $f_2 = \bar{x}(yz + \bar{y})$

ES 2.1

$n1 = (F3)_{16}$
$n2 = (71)_{16}$
$(C2)$

$(1111\ 0011)$ → $-128+64+32+16+3 = (-13)_{10}$
$(0111\ 0001)$ → $(113)_{10}$

$1\ 0110\ 0100$ no overflow.

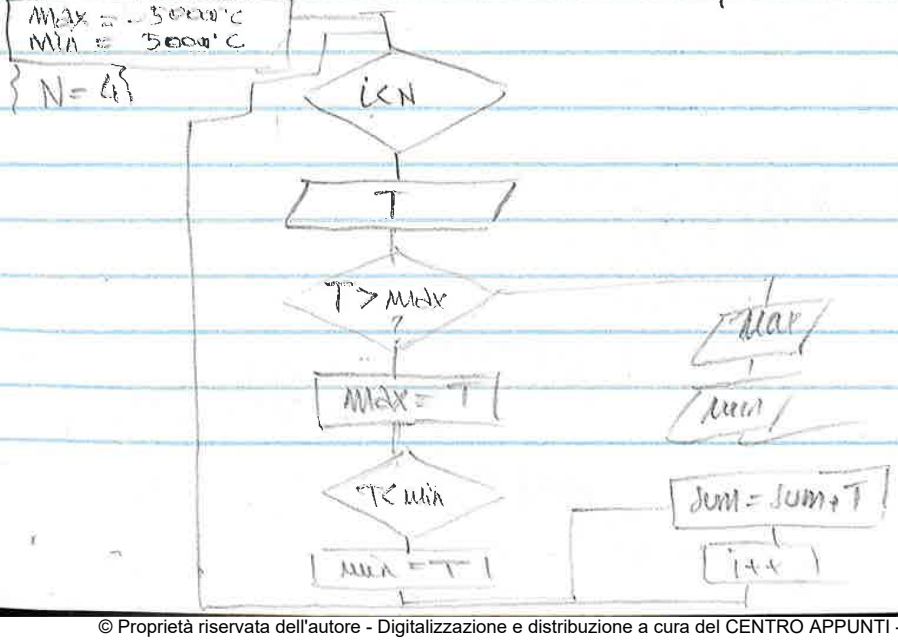
ES 2.2

$f_1 = xy\bar{z} + z\bar{y}$, $f_2 = \bar{x}(yz + \bar{y})$

x	y	z	f1	f2
0	0	0	0	1
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	0

ES 1. • Leggere da tastiera delle temperature (int.)
- T max, - T min, T media, quelle T sono comprese in un certo intervallo [n1, n2] da tastiera
• Fase di esercizio sia con #define che con interruzione fino a T = -1000°C

SUM = 0



$media = \frac{Sum}{N}$ (float)

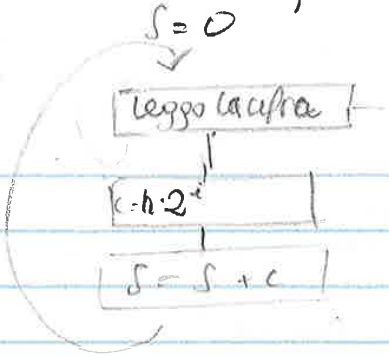
14. ES: Leggere un numero binario e calcoliamo quanto vale in decimale

→ 100110

2^5 2^4 2^3 2^2 2^1 2^0

$= 32 + 4 + 2 = (38)_{10}$

n volte quanti sono i bit




```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define N 6

int main () {
    int n, s, i;
    s = 0;
    for (i=0; i<N; i++) {
        printf ("Bit = ");
        scanf ("%d", &n);
        if ((n!=0) && (n!=1)) {
            printf ("Error No Binary");
            break;
        }
        s = s + n * pow(2,i);
    }
    printf ("Decimale = %d", s);
    return 0;
}
```

ok

controllo cifra inserita

ES: Serie di fibonacci 0, 1, 1, 2, 3, 5, 8, 1, , , ...

$x_i = x_{i-1} + x_{i-2}$

1 = 0+1

prec : succ

succ = somma

```
#include <stdio.h>
#include <stdlib.h>
#define N 9
int main () {
    int i, somma, prec, succ;
    prec = 0;
    succ = 1;
    printf ("Serie di Fibonacci: \n");
    for (i=0; i<N; i++) {
        somma = prec + succ;
        prec = succ;
        succ = somma;
        printf ("%d\n", somma);
    }
    return 0;
}
```

Soluzione con for . Non uso le librerie matem

```
#include <stdio.h>
#include <stdlib.h>
int main ( ){
    int ci, c, a, i, p;
    float r;
    printf ("CAPITALE INIZIALE = ");
    scanf ("%d", &ci);
    printf ("RENDIMENTO = ");
    do {
        scanf ("%f", &r);
    }
    while ( r<1);
    printf ("NUMERO ANNI = ");
    scanf ("%d", &a);
    p = ci;
    for (i=1; i<=a; i++) {
        c = p*(1+r);
        p=c;
    }
    printf ("CAPITALE FINALE = %d", c);
    return 0;
}
```

OK

ES radice $\sqrt{N}$

1) x

2) $x_n = \left(x_{n-1} + \frac{N}{x_{n-1}}\right)\cdot\frac{1}{2}$ → quando mi fermo? $\varepsilon \le |x_n - x_{n-1}| = 0.01$

$x_0 = \frac{N}{2}$

Da tastiera leggo N ed ε

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main () {
    int N, i=0;
    float x, p, d, eps;
    p = (float) N/2;
    d = eps + 1;
    printf ("Numero = ");
    scanf ("%d", &N);
    printf ("Errore = ");
    scanf ("%f", &eps);
    while (d > eps) {
        x = (p + N/p) * 0.5;
        d = x - p;
        p = x
        i++;
    }
    printf ("Radice = %d In Iterazioni = %d", x, i);
    return 0;
}
```

OK

16 Se non conosco il numero di cicli a priori in un ciclo, posso usare il flag:

es → for(i=0; flag==0; i++) {

} ! evento → flag=1

Leggiamo da tastiera coppie di numeri n1,n2. Calcolare la coppia $|n_1^2+n_2^2|$ è massimo. Fermarsi quando $n_1+n_2=0$ oppure dopo N volte.

flag=0 (ripeti acquisisci n1,n2)

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int n1,n2, flag=0;
    int K, S, max=0;
    for (i=0; flag==0; i++) {
        printf("Coppia %d = ", i);
        scanf("%d \n %d", &n1, &n2);
        if (n1+n2==0) {
            flag=1;
        }
        S = (n1)^2 + (n2)^2;
        if (S > max) {
            max=S;
            K=i;
        }
    }
    printf("Coppia MOD MAX = %d", K);
    return 0;
}
```

< VETTORI >

Insieme di variabili ordinate dello stesso tipo

Es int vettore[10] → Lettura e scrittura con il ciclo for.

Es. Leggere da tastiera N vett. di 3 elementi e calcolare:
- prodotto solo pari
- somma dei valori

```
#include <stdio.h>
#include <stdlib.h>
#define N 3
int main() {
    int vett[N];
    int i, S=0, pari=0;
    printf("Vettore = ");
    for (i=0; i<N; i++) {
        scanf("%d", &vett[i]);
        S = S + vett[i];
        if (vett[i]%2==0) {
            pari++;
        }
    }
    printf("Elem. pari = %d \n
        Somma = %d",
        pari, S);
    return 0;
}
```

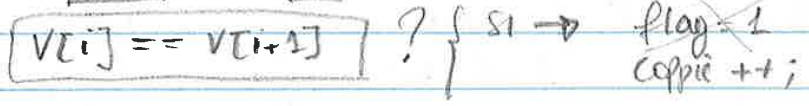
Leggere da tastiera per n giorni piovosità di una settimana → vettore.
- piovosità media
- n° giorni in cui piovosità > soglia
- diff tra giorno + piovoso e gg - piovoso

```
#include <stdio.h>
#include <stdlib.h>
#define N 7
#define SOGLIA 100.
int main () {
    int rain[N];
    int is = 0, i, min, max, d, S=0;
    float media;
    max = -100000;
    min = 100000;
    printf (" INDICI PIOVOSITA' SETTIMANA : \n");
    for (i=0; i<N; i++) {
        scanf (" %d ", &rain[i]);
        S = S + rain[i];
        if ( rain[i] > soglia ) {
            is++;
        }
        if ( rain[i] > max ) {
            max = rain[i];
        }
        if ( rain[i] < min) {
            min = rain[i];
        }
    }
    d = max-min;
    media = S/N;
    printf (" MEDIA = %d \n GG OLTRE SOGLIA = %d \n DIFF = %d \n",
                media, is, d );

    return 0;
}
```

VETTORE n elementi → quante coppie di elementi adiacenti sono uguali?

Es: 0, 1, 2, 2 n coppie = 1.




```
#include <stdio.h>
#include <stdlib.h>
#define N 7
int main () {
    int V[N], i, coppie = 0;
    printf (" vettore = \n");
    for ( i=0; i<N; i++) {
        scanf (" %d ", &V[i]);
    }
    for ( i=0; i<N; i++) {
        if ( V[i] == V[i+1] ) {
            coppie ++;
        }
    }
    printf (" coppie = %d ", coppie);
    return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#define N 6
int main(){
    int V[N]={5, 1, 3, -1, 7, 6};
    int i,J, num, cont = 0;
    printf ("NUMERO = ");
    scanf ("%d", &num);
    printf ("VETTORE = );
    for (i=0; i<N; i++){
        for (J= i+1; J<N; J++){
            if (V[i]+V[J]==num){
                cont++;
                printf ("%d - %d\n"
                        V[i], V[J]);
            }
        }
    }
    printf ("La somma si trova %d volte.", cont);
    return 0;
}
```

Es] V1[N], V2[M] → letto da tastiera. Quanti elementi di V1 sono presenti in V2?

```
#include <stdio.h>
#include <stdlib.h>
#define N 6
#define M 4
int main(){
    int V1[N]={5,1,3,-1,7,6};
    int V2[M], i,J, cont=0;
    printf ("V2=");
    for (i=0; i<M; i++){
        scanf ("%d", &V2[i]);
    }
    for (i=0; i<M; i++){
        for (J=0; J<N; J++){
            if (V2[i]==V1[J]){
                cont++;
            }
        }
    }
    printf ("%d Elementi di V1 presenti in V2.", cont);
    return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#define N 6

int main () {
    int V[N], i, j, t;
    int V[N] = { 67, 3, 5, 12, 7, 4};

    for (i=0; i<N-1; i++) {
        for (j = i+1; j<N; j++) {
            if (V[i] < V[j]) {
                t = V[i];
                V[i] = V[j];
                V[j] = t;
            }
        }
    }
    printf ("VETTORE ORDINATO = ");
    for (i=0; i<N; i++) {
        printf ("%d \n", V[i]);
    }
    return 0;
}
```

OK

ES 1. Leggere dei num. da tastiera e li mette in due vettori V1 e V2. In V1 tutti pari, in V2 tutti dispari. [] di max N
Genero un terzo vettore che contiene tutti gli elem. di V1 e di V2 ordinati.

```
#include <stdio.h>
#include <stdlib.h>
#define N 4

int main () {
int V1[N], V2[N], W[2*N], n, i, K=0, J=0, L=0, t;
while ((K<N) && (J<N)) {
    scanf ("%d", &n);
    W[L] = n;
    L++;
    if (n%2 == 0) {
        V1[i] = n;
        i++;
    }
    else {
        V2[J] = n;
        J++;
    }
}
printf ("V1 = \n");
for (i=0; i<K; i++) {
    printf ("%d \n", V1[i]);
}
printf ("V2 = \n");
for (i=0; i<J; i++) {
    printf ("%d \n", V2[i]);
}
printf ("W = \n");
for (i=0; i<L; i++) {
    printf ("%d \n", W[i]);
}

for (i=0; i<L-1; i++) {
    for (J= i+1; J<N; J++) {
        if (W[i] < W[J]) {
            t = W[i];
            W[i] = W[J];
            W[J] = t;
        }
    }
}
```

// Ordine decrescente //

```
printf ("decrescente = \n");
for (i=0; i<L; i++) {
    printf ("%d \n", W[i]);
}
```

```
#include <stdio.h>
#include <stdlib.h>
#define N 5
#define M 3
int main() {
    int V1[N] = {0,5,-3,2,9}, V2[M] = {5,-3,2}, cont = 0, i,J, pos;
    for (i=0; i<N; i++) {
        if (V1[i] == V2[0]) {
            pos = i;
        }
    }
    for (i=pos; i<N; i++) {
        for (J=0; J<M; J++) {
            if (V1[i] == V2[J]) {
                cont++;
            }
        }
    }
    if (cont == M) {
        printf ("vettore V2 contenuto in V1");
    }
    else {
        printf ("vettore V2 non contenuto in V1");
    }
    return 0;
}
```

OK

[Es] Per ogni giorno della settimana valutiamo la Tmax (float) e l'ora (int).

- Calcolare l'escursione max (differenziale max delle t)
- Ordinarli.

Calcolo la differenza di ogni elemento con tutti gli altri

ex = -10000

i : 0 ÷ N-1
J : i+1 ÷ N

```
if ((T[i]-T[J])>ex) {
    ex = T[i] - T[J];
    J++;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#define N 5
int main() {
    int h[N], i,J, tm;
    float t[N], ex = -10000;
    printf ("Temperature e orari: \n");
    for (i=0; i<N; i++) {
        printf ("T = ");
        scanf ("%f", &t[i]);
        printf ("h = ");
        do {
            scanf ("%d", &h[i]);
        }
        while ((h[i]<0) || (h[i]>24));
    }
```

ESEMPIO Creare una funzione "PARI" a cui passo un numero intero e mi ritorna : 0 => PARI
1 => DISPARI

Il programma legge da tastiera 3 numeri e mi dice se è pari o dispari

```
int pari (int n) {
    int r;
    if ((n%2)==0) {
        r=0;
    }
    else {
        r=1;
    }
    return r;
}
```

CORPO

```
#include <stdlib.h>
#include <stdio.h>
#define M 3
int pari (int n);
int main () {
    int a, c, i;
    for (i=1; i<=M; i++) {
        printf ("NUMERO %d = ", i);
        scanf ("%d", &a);
        c = pari (a)
        printf ("%d\n", c);
    }
    return 0;
}
int pari (int n) {
    int r;
    if ((n%2)==0) {
        r=0;
    }
    else {
        r=1;
    }
    return r;
}
```

ok

ES pow(b,e) funzione expo(tipo powel) che riceve 2 valori a e b
Leggere 2 numeri da tastiera.

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

NB esponenziale : a^b:

```
ris=1;
for (i=0; i<b; i++) {
    ris = ris*a;
}
```

```
#include <stdio.h>
#include <stdlib.h>
int expo (int a, int b);
int main () {
    int p, q, ris;
    printf ("BASE = ");
    scanf ("%d", &p);
    printf ("ESPONENTE = ");
    scanf ("%d", &q);
    ris = expo (p, q);
    printf ("exp = %d", ris);
}
int expo (int a, int b) {
    int e, i;
    e = 1;
    for (i=0; i<b; i++) {
        e = e*a;
    }
    return e;
}
```

ok

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
float lenght ( int a, int b, int c, int d);
int main () {
        int   x1, x2, y1, y2, x3, x4, y3, y4;
        float L12, L34;

        printf ("P1 = \n");
        printf ("x1 = ");
        scanf ("%d", &x1);
        printf ("y1 = ");
        scanf ("%d", &y1);
                                        } x 4  // acquisisco le coordinate //
                .
                .
                .
        L12 = lenght (x1, y1, x2, y2);
        L34 = lenght (x3, y3, x4, y4);
        printf ("L12 = %f ____ L34 = %f \n", L12, L34);
        if (L12 >= L34)
                printf ("Max P1P2");
        else    printf ("Max P3P4");
        return 0;
}
float lenght (int a, int b, int c, int d) {
        float m;
        m = sqrt ( (a-c) * (a-c) + (b-d) * (b-d) );
        return m;
}
```

PROBLEMA: funzioni che ritornano + di un valore ⟶ PUNTATORI

"Scatole" che non contengono dei valori, ma degli indirizzi.

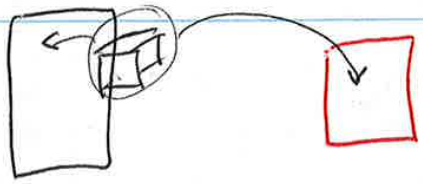
1) `int *p;` // all'interno di "p" vado a mettere indirizzo di altra variabile, in questo caso intera //
   `int a;`

⇒ p = &a (indirizzo di a)

*p = 3 ⟺ a = 3 // Metto "3" nella scatola il cui indirizzo è in p, quindi in p c'è l'indirizzo di "a". e porta "3" in "a" //

Es:
```
int *p;
int x;
p = &x;   <- Da questo momento in poi *p "=" x
```

↳ CONDIVISIONE della "SCATOLA" fra main e funzione:



passo alla funzione l'indirizzo della SCATOLA

1ES) Dato un vettore, scrivere una funzione che ritorna:
- il valore del max;
- il valore del min;
- Varianza;

$$\sigma = \sqrt{\frac{\sum (x_i - x_M)^2}{N}}$$

2°ES) Un radar scandisce una linea di N elementi. Ogni elemento è un punto dello spazio e vale 0 se non c'è un ostacolo e 1 se c'è un ostacolo.

N | 0 0 1 1 0 1 0 |

Ostacolo Ostacolo (adiacenza di 1)

a) n° ostacoli
b) l'ostacolo + grande (lunghezza)

1) VARIANZA: $sqs = sqs + (x_i - x_M) * (x_i - x_M)$; $sqs = 0$

$VZ = sqrt(sqs/N)$

V = {2, 5, -5, 8, 0}

MAX = 8
MIN = -5
media =

$$Var = \sqrt{\frac{3^2 + 1^2 + 7^2 + 6^2}{5}} = \sqrt{\frac{9 + 49 + 36}{5}} = \sqrt{18,8} = 4,33$$

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define N 5
float f ( int V[], int M, *a, *b);

int main ( ) {
    int W[N] = {2,5, -5,8,0}, max, min, v
    float var;
    var = f ( W, N, &max, &min);
    printf ( "MAX = %d ____ MIN = %d ____ VARIANZA = %f", max, min, var);
    return 0;
}

float f (int V[], int M, *a, *b){
    int mx = -10000, mn = 10000, i, s=0, sqs=0;
    float md, vz;
    for (i=0; i<N; i++) {
        if (V[i] > mx)
            mx = V[i];
        if (V[i] < mn);
            mn = V[i];
        s = s + V[i];
    }
    md = (float) s/M;
    for(i=0; i<N; i++){
        sqs = sqs + (V[i]-md) * (V[i]-md);
    }
    vz = Sqrt ( sqs/M);
    *a = mx;
    *b = mn;
    return vz;
}
};
```

LEZ 25 <CARATTERI> gestiti dal codice ASCII.

- Ogni carattere è rappresentato dal suo codice ASCII;

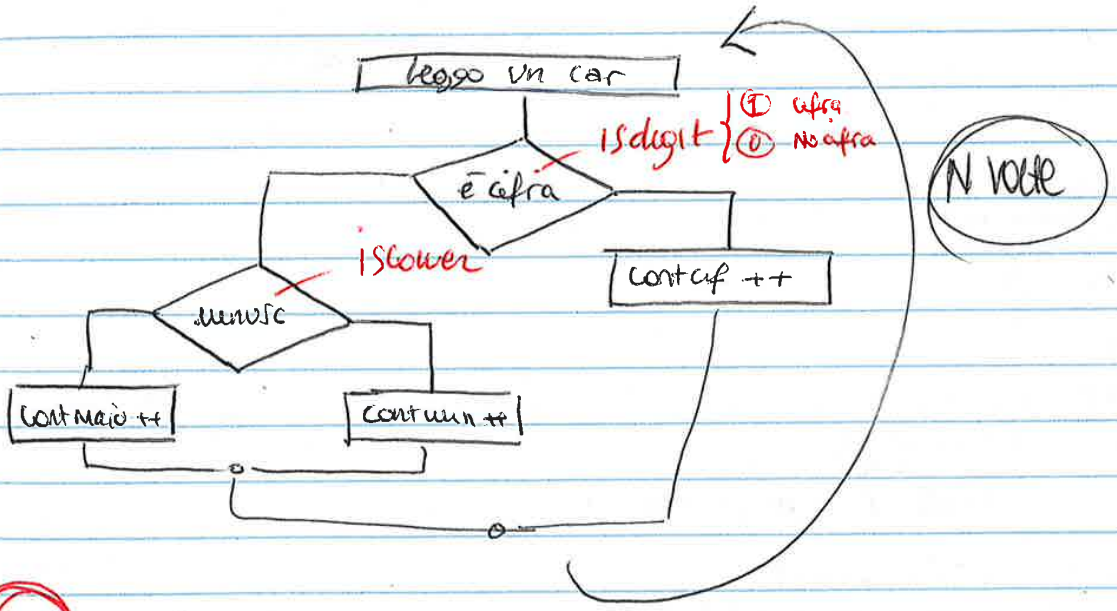
Leggere un carattere: { scanf("%c", car); / getchar(car);

Stampare un carattere: { printf("%c", car); / putchar(car);

FUNZIONI DI UTILITA' (dalla Libreria <CTYPE.h>

int isalpha(char c)	ci dice se c è lettera;
int isdigit(char c)	" " " " cifra;
int islower(char c)	" " " " minuscola;
int isupper(char c)	" " " " maiuscola;
int tolower(char c)	Converte in minuscolo;
int toupper(char c)	Converte in maiuscolo.

ESEMPIO. Leggiamo da tastiera 4 caratteri e ci chiediamo se sono cifre o lettere, se sono lettere quali maiuscole e quali minuscole.



NB: Quando gli faccio leggere un carattere lui considera anche "invio" per andare a capo. La scrittura:

scanf("%c%*c", &c)

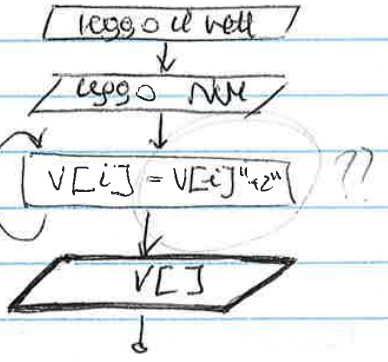
Legge 2 caratteri e il secondo lo butta via.

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
int main(){
    int i, M=0, m=0, num=0;
    char c;
    for(i=0; i<5; i++){
        printf("carattere =");
        scanf("%c%*c", &c);
        if(isdigit(c)!=0)
            num++;
        else{
            if(islower(c)!=0)
                min++;
            else
                M++;
        }
    }
    printf("NUM = %d \n MAIUSCOLE = %d \n MIN = %d",
        num, M, m);
    return 0;
}
```

Es Cifrare un testo con l'algoritmo di Cesare.

Marco → (n=2) ogni lettera è spostata di 2.
⇒ {octeq}
Algebra modulo 26

vettore V[] = {__}
N



M = 26

J | a b c d e f g h i j k l m n o p q r s t u v w x y z |

i | marco | N = 5

'm' == 'a' NO → 'm' == 'b'

testo[0] ↔ alfabeto[0] 0
0 - N alfabeto[1] M
alfabeto[26]

{! w a b y 3} → {!gkli3}

NB: Se l'alfabeto "finisce", deve ricominciare da capo
↳ M - J < key

ESEMPIO: V[] = {qui} key = 10
↳ {aes}

q → i = 19
26 - 19 = 7
↳ alfabeto[ ]
key - (M - J)

quive → aesfo
⊕ controllo alfabetico

OK

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define N 5
#define M 26

int main() {
    char abc[M] = {'a','b','c','d', .... 'y','z'}, tx[N] = {'m','a','r','c','o'};
    int i, J, key = 10;
    for (i=0; i<N; i++) {
        if (isalpha(tx[i])) {
            for (J=0; J<M; J++) {
                if (tx[i] == abc[J]) {
                    if ((M-J) > key)
                        printf("%c", abc[J+key]);
                    else printf("%c", abc[key-M+J]);
                }
            }
        }
        else printf("%c", tx[i]);
    }
    return 0;
}
```

ESEMPI | Leggere una stringa da tastiera:
- Stampare la lunghezza;
- Contare quanti caratteri sono cifre;
- Trasformare tutte le maiuscole in minuscole.

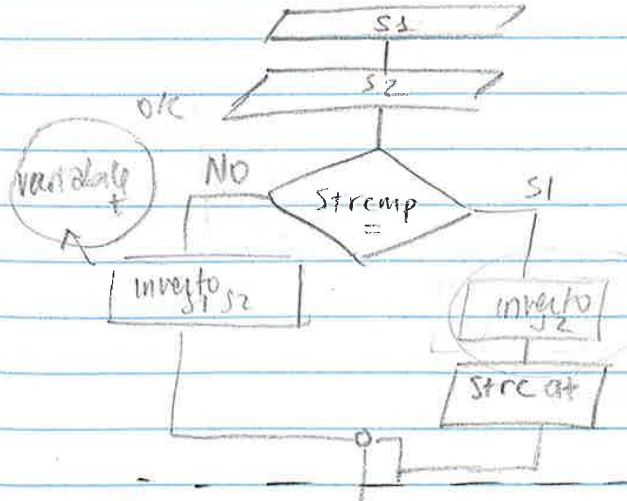
} per valutare questi punti mi recordo che una stringa è un vettore di caratteri

for

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#define N 100

int main(){
    char s[N+1];
    int i, cont=0;
    printf("Stringa=");
    gets(s);
    for(i=0; i<N; i++){
        if(isdigit(s[i]))
            cont++;
        if(isupper(s[i]))
            s[i]=tolower(s[i]);
    }
    printf("LUNGHEZZA = %d \t CIFRE = %d \n", strlen(s), cont);
    printf("STRINGA = %s", s);
    puts(s);
    return 0;
}
```

Es| s_1 ed s_2 lette da tastiera. Se le stringhe sono uguali, inverto i caratteri in s_2 e poi concateno. Se sono ≠ inverto s_1 ed s_2.

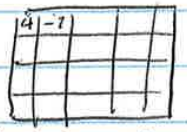



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define N 100
int main(){
char s1[2*N+1], s2[N+1], t, z[N+1];
int i,j;
printf("STRINGA=");
gets(s1);
printf("STRINGA=");
gets(s2);
if(strcmp(s1,s2)==0){
    j=0;
    for(i=strlen(s2)-1; i>=0; i--){
        z[j]=s2[i];
        j++;
    }
    printf("STRINGA=%s", strcat(s1,z));
}
else{
    for(i=0; i<N; i++){
        t=s1[i];
        s1[i]=s2[i];
        s2[i]=t;
    }
    printf("STRINGA=%s \t STRINGA=%s", s1, s2);
    return 0;
}
```

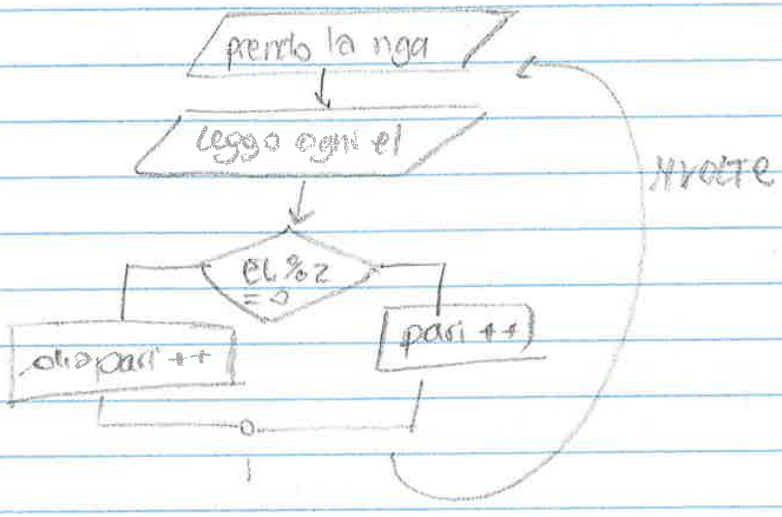
<MATRICE> ⇔ vettore multidimensionali

ES: int MATRICE [N] [M] → righe, colonne

```
for (i=0; i<N; i++){
    for (j=0; j<M; j++) {
        // operazioni //
        :
    }
}
```



ES Matrice [R] x [C]
calcolare quanti sono gli elementi pari e dispari di ciascuna riga e colonna.




```
#include <stdio.h>
#include <stdlib.h>
#define R 3
#define C 4

int main() {
    int m[R][C] = {{-3,5,6,9},{17,4,2,9},{45,89,1,65},{24,32,6,0}}, i, j,
        pari, dispari, p[R], d[R];

    printf("RIGHE:\n");
    for (i=0; i<R; i++) {
        pari = 0;
        dispari = 0;
        for (j=0; j<C; j++) {
            if (m[i][j] % 2 == 0)
                pari++;
            else
                dispari++;
        }
        p[i] = pari;
        d[i] = dispari;
    }
    printf("Num pari \ riga:\n");
    for (i=0; i<R; i++) {
        printf("%d ", p[i]);
    }
    printf("Num dispari / colonna\n");
    for (i=0; i<R; i++) {
        printf("%d ", d[i]);
    }
    return 0;
}
```

OK

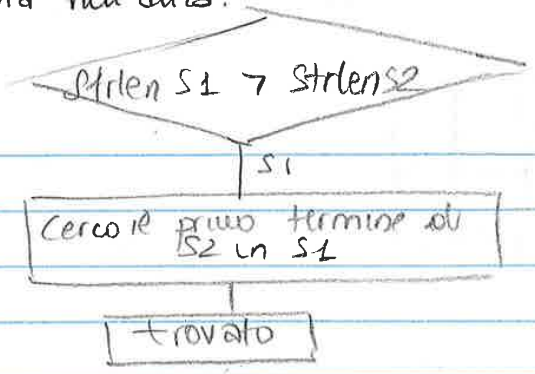
NB Per colonne basta sostanzialmente invertire le posizioni dei 2 cicli for

Es) L'utente inserisce due parole [al max 30] da tastiera e il programma deve dire se una è contenuta nell'altra:

s1 | a | l | + | r | e | \0 |

s2 | t | r | e | \0 |

* Strlen non conta '\0' *



s1 tre

s2 altre

lez 29

ESJ Matrice di interi RxC. Leggere da tastiera: n r c
e calcolare la somma della riga o della colonna corrispondente. (funzione)

0	5	4	2
1	6	7	8
0	0	1	4

R=3
C=4

Leggo 2 c ⇒ Σ elem. 2ª colonna

le colonne vanno dichiarate esplicitamente

!! Passare una matrice a una funzione:

int f (int m[][C], int RR, char a, int b)

mi passa la matrice,

```
# include <stdio.h>
# include <stdlib.h>
# define R 3
# define C 4
int f ( int M[][C], int RR, char a, int b);
int main() {
        int m[R][C] = {{0,1,2,3},{0,0,4,1},{5,8,0,1}}, n, somma;
        char rc;
        printf ("Riga o colonna: ");
        scanf ("%*c%c", &rc);
        printf ("Valore: ");
        scanf ("%d", &n);
        somma = f (m, R, rc, n);
        printf ("somma = %d", somma);
        return 0;
}
int f ( int M[][C], int RR, char a, int b) {
        int i, sum = 0;
        if (a == 'r') {
                for (i = 0; i < RR; i++)
                        sum = sum + M[a][i];
        }
        else if ( a == 'c') {
                for ( i=0; i<C; i++)
                        sum = sum + M[i][a];
        }
        return s;
}
```

ok

1) APERTURA FILE

```
FILE *fp
fp = fopen ("nome.txt", "r")
```

2) LEGGERE IL FILE

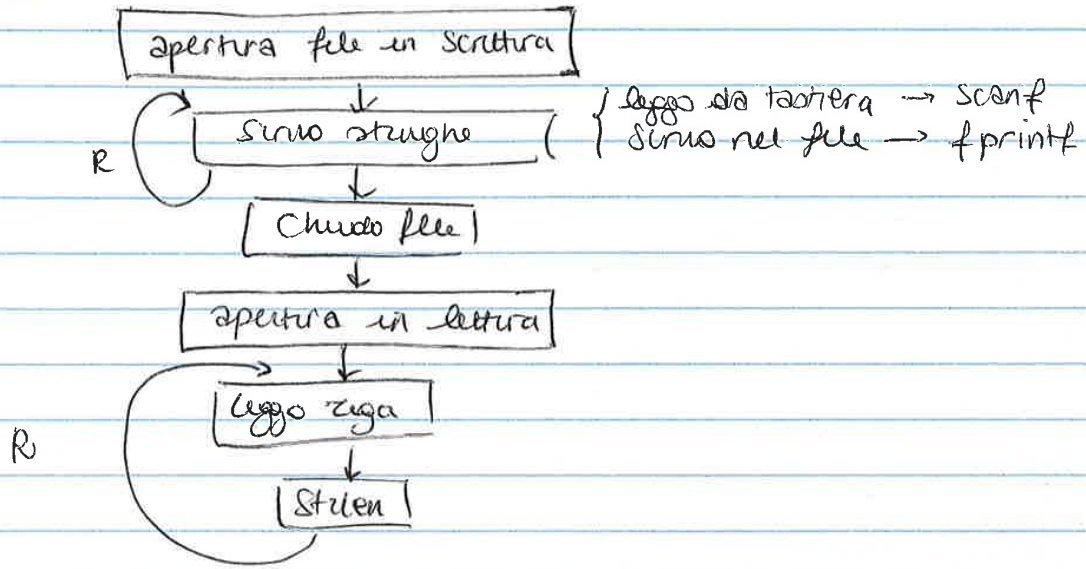
`fscanf (fp, "%s", s1)` legge una riga del file

3) SCRIVERE IL FILE

`fprintf (fp, "%...", ...)` scrive una riga del file

\n

Scrivere un programma che legge da tastiera R stringhe, genera un file con le stringhe lette, rileggo il file e stampa a video di quanti elementi è composto.




```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define R 3
#define C 20
int main () {
    char s[C];
    int i; FILE *fp
    fp = fopen("data.txt", "w");
    if (fp == NULL) {
        printf("Error\n");
        return -1;
    }
```

→ // dopo l'apertura, faccio il controllo //

(ok)

```
    else { for (i=0; i<R; i++) {
        printf ("stringa = ");
        gets(s);
        fprintf (fp, "%s\n", s);
    } fclose (fp); }
    fp = fopen ("data.txt", "r");
    if (fp == NULL) {
        printf ("Error\n");
        return -1;
    } else {
        for (i=0; i<R; i++) {
            fscanf (fp, "%s", s);
            printf ("L(%d) = %d", i, strlen(s));
        }
        fclose (fp);
    }
    return 0;
}
```

```
        max = n;
        posmax = i;
        Strcpy ( cittamax, S);
        }
    i++;
}
printf ("Righe lette=%d  Località Max = %s, (%d)%d", cittamax, max, posmax);
fclose (fp);
return 0;
}
```

OK

Es Un file contiene una serie di righe: $ig = 0.8V1 + V2 + 0.5V3$

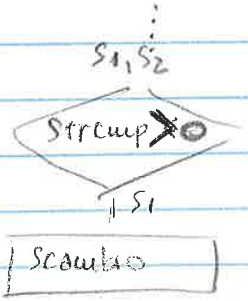
<codice> <V1> <V2> <V3>

S[] int int int

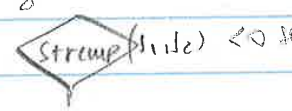
Il programma deve ordinare i pezzi:
- in ordine alfabetico + ig
- ig decrescente

Il numero di righe non può eccedere ROW ma non è noto.

- codice 1 — 1 2 3
- cod 02 — 4 5 6
- cod 03 — 6 7 8



Ordine alfabetico + confronto 2 righe:




```
for(i=0;...){
    for(j=i+1;...){
        if(Strcmp)
        //scambio
```

1a riga → col tutti gli altri
2a riga → col restanti
⋮
penultima riga → col restante

OSS.
- Negli esercizi in cui devo fare un confronto fra righe, come l'ordinamento, ho bisogno di leggere PRIMA tutti gli elementi del file su cui lavorare
- se devo invece fare delle operazioni sottostanti su ogni riga posso farlo "durante" la lettura.
- "Prelevate le informazioni che mi servono, posso chiudere il file.

TEMA D'ESAME OL. Ogni elemento di una matrice N x N indica la pressione relativa ad un frammento di territorio. Si vogliono determinare le posizioni "critiche", cioè
- un'area a bassa prex circondata da aree ad alta prex
- " " " alta " " " " " " " bassa "

- La matrice è scritta in un file
- La matrice di rischio è data da una 3x3

m[N][N]

6	5	4	10
4	10	8	9
3	2	1	9
10	4	5	6

Parto da 0,0:

```
for (i=0; i<N-3; i++) {
  for (j=0; j<N-3; j++) {
    for (h=i; h<i+3; h++) {
      for (k=j; k<j+3; k++) {
centrale = m[h+1][k+1]
```

confronto elemento centrale con quelli attorno.

```
if ( m[h][k] >= centrale )
    flag = 1;
}
}
if (flag == 1) {
//zona a rischio//
```

```
#include <stdio.h>
#include <stdlib.h>
#define N 6
int main () {
```

```
i=0;
while ( fscanf (fp, "%d %d %d %d", &m[i][0], &m[i]
```

fo

da controllare

" centrale "

ES] Il programma prende in ingresso da l.c. il nome del file e un numero (+1) → argc = 3

Visualizzare a video gli argomenti:

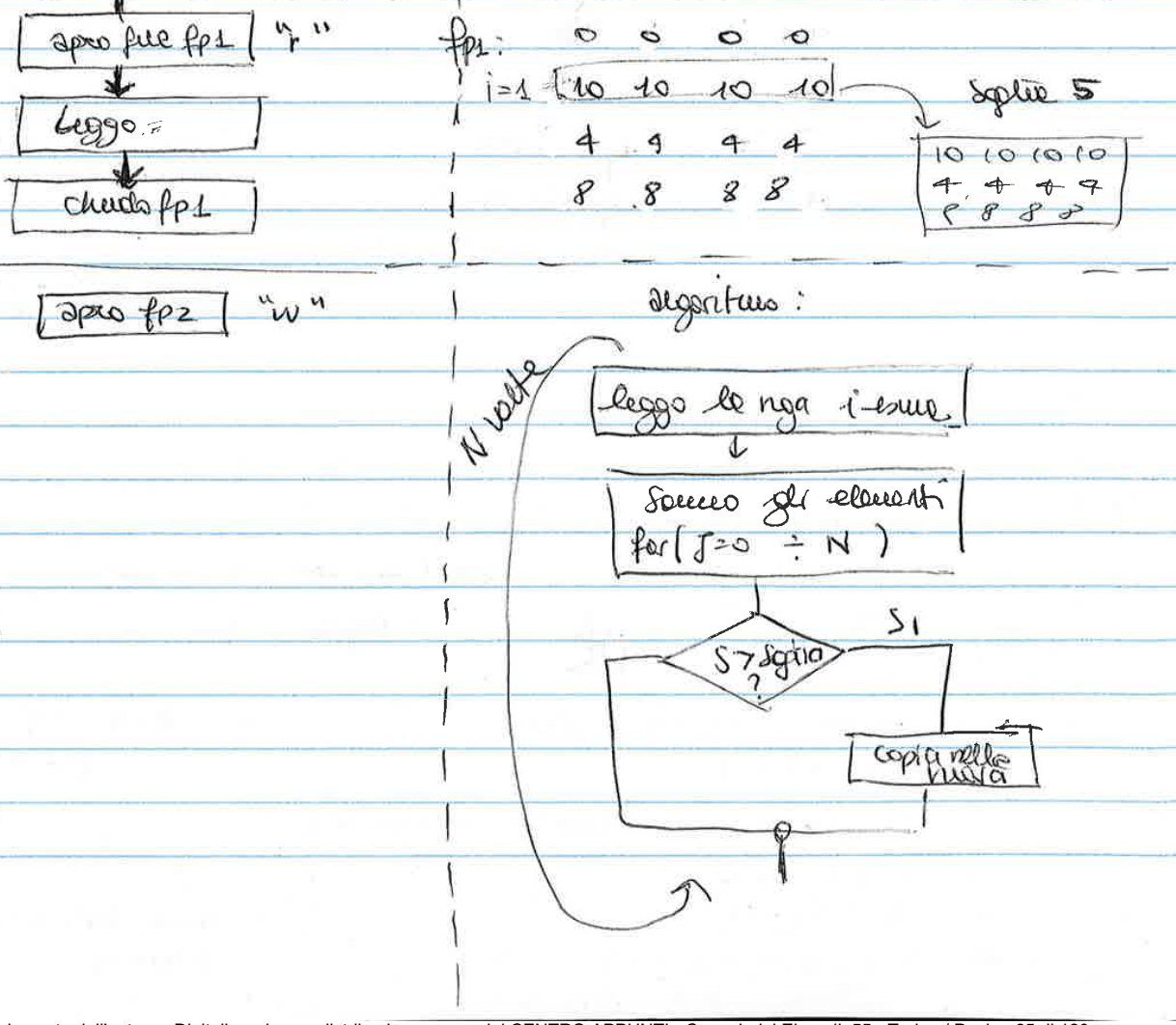
ok

```
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[]) {
    int i;
    if (argc != 3) {
        printf ("Errore linea di comando\n");
        return -1;
    }
    printf ("Argomenti = %d \n", argc);
    for (i=0; i<argc; i++) {
        printf ("%s \n", argv[i]);
    }
    return 0;
}
```

[ES] Un file contiene una matrice numerica N×N. <TIPO ESAME>

1) Leggere la matrice → MM[][]
2) Generare un file che contiene una matrice di M[][] che contiene le righe di MM / Σ elem > data soglia. (stesso numero di colonne, il max diventa il numero di righe)
   ↳ 3) la scriviamo in un file.

Da linea di comando nome del file e la soglia.



ES: matrice con C colonne e al + R righe. <TIPO ESAME>

(char)

P	M	P	M
M	P	M	M
P	P	M	M
P	P	P	M

r=2
c=1

Da tastiera leggo una posizione → 2 coordinate
- se è montagna, esce
- se è pianura, quante P ci sono sopra, sotto, a dx e a Sx

NB La matrice di caratteri si legge riga x riga nel file
fscanf(fp, "%s", m[i])

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#define C 4
#define RR 7
int main () {
    int i, x, c, R, destra = 0, sinistra = 0, sopra = 0, sotto = 0;
    char t[RR][C];
    FILE *fp;
    fp = fopen ("ter.txt", "r");
    if (fp == NULL) {
        printf ("errore apertura\n");
        return 0;
    }
    i = 0;
    while (fscanf (fp, "%s", t[i]) != EOF) {
        i++;
    }
    R = i
    printf ("Righe lette = %d\n", R);
    printf ("Terreno : \n");
    for (i = 0; i < R; i++) {
        puts (t[i]);
    }
    fclose (fp);
    printf ("coordinata riga = ");
    scanf ("%d", &x);
    printf ("coordinata colonna = ");
    scanf ("%d", &c);
    if (t[x][c] == 'm') {
        printf ("montagna\n");
        return -2;
    }
    else {
        for (i = c+1; i < C; i++) {
            if (t[x][i] == 'p')
                destra++;
        }
        for (i = c-1; i >= 0; i--) {
            if (t[x][i] == 'p')
                sinistra++;
        }
        for (i = x+1; i < R; i++) {
            if (t[i][c] == 'p')
                sotto++;
        }
        for (i = x-1; i >= 0; i--) {
            if (t[i][c] == 'p')
                sopra++;
        }
    }
}
```

Es. Definisco con una struttura i punti nello spazio 3D.
Un file è composto da tot righe; ogni riga contiene 3 interi che sono le coordinate di un punto. Calcolare il Baricentro.

$$G \begin{cases} x_B = \frac{\sum x_i}{N} \\ y_B = \frac{\sum y_i}{N} \\ z_B = \frac{\sum z_i}{N} \end{cases}$$

3 6 12 4 0 2 3 3 3
3 6 12 7 6 8 6 6 6
3 6 12 4 3 5 9 9 9
3 6 12

```
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[]) {
    int i, R, sumx=0, sumy=0, sumz=0;
    struct punto {
        int x;
        int y;
        int z;
    };
    struct punto p;
    if (argc!=2) {
        printf ("Errore l.c.\n");
        return -1;
    }
    FILE *fp;
    fp= fopen ("coord.txt","r");
    if (fp == NULL) {
        printf ("Errore Apertura\n");
        return -2;
    }
    R=0;
    printf ("Contenuto file :\n");
    while (fscanf (fp, "%d %d %d", p.x, p.y, p.z)!= EOF) {
        R++;
        sumx += p.x;
        sumy += p.y;
        sumz += p.z;
        printf ("%d %d %d\n", p.x, p.y, p.z);
    }
    fclose(fp);
    printf ("Regno letto = %d\n Coordinate Baricentro : (%f, %f, %f)", R,
        (float) sumx/R; (float) sumy/R ; (float) sumz/R);
    return 0;
}
```

```
#include <Stdio.h>
#include <stdlib.h>
#define N 20

int main (int argc , char *argV[]){
    char cod[20], g[2];
    int i,R, t, tag[N], T={38,40,42,44,46,48,50,52,54,56,58}, tlc, trovato=0, cont=0;
    float p, valore=0;

    if (argc !=4) {
        printf ("Errore l.c.\n");
        return -1;
    }
    FILE *fp;
    fp = fopen ("reg.txt", "r");
    if (fp==NULL) {
        printf ("Errore Apertura\n");
        return -2;
    }
    R=0
    for (i=0; i<N; i++) {
        tag[i]==0;
    }
    tlc = atoi (argv[3]);
    printf (" Negozio : \n");
    while ( fscanf (fp, "%s %s %d %f", cod, g, &t, &p) != EOF ) {
        tag[R] = t;
        R++;
        valore += p;
        printf ("%s %s %d %f\n", cod, g, t, p);
        if(( t== tlc) && (g[0] == argV[2][0] ))
            trovato++;
    }
    printf (" Quanti negozio = %d \n Valore totale = %f \n Capi venduti da l.c. = %d\n",
            R, valore, trovato);
    fclose(fp);
    for(i=0; i<11; i++) { cont=0;
        for( J=0; J<R; J++) {
            if ( tag[J] == T[i])
                cont++;
        }
        printf ("Trovate %d taglie %d\n", cont, T[i]);
    }
    return 0;
}
```

OK

Per trovare l'opposto di un numero in CA2:

- Inverto BIT a BIT
- Sommo 1

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 71 di 120

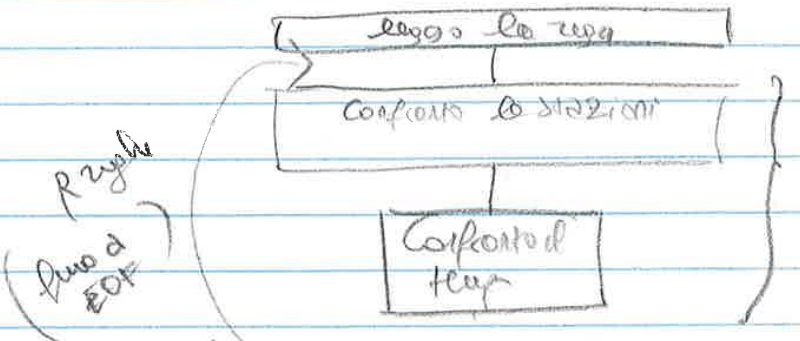
Domande:

- Cosa sono e a cosa servono i bus? (Collegano la memoria alla CPU)
  - Sono una serie di collegamenti che costituiscono il "sistema circolatorio" del pc. Sono addetti al trasporto e al controllo dei dati
    - ABUS "address bus" di indirizzo
    - DBUS "Data bus" dei dati
    - CBUS "Control bus".
- MAX memoria se ho ABUS = 16 Bit? → 2^{16} byte → 64 KByte.
  MAXMEM = $(2^{ABUS}) \cdot DBUS$
- C, funzione → By value (passo il valore)
  → By reference (passo l'indirizzo tramite il puntatore)
- ALU: Aritmethic and logic unit è presente nella CPU ed è addetta all'esecuzione di calcoli e operazioni matematiche
- Che cos'è il clock e come si misura: è il temporizzatore della CPU. Si misura in Hz

<PROGRAMMAZIONE> Ogni riga di un file contiene

< S. partenza > < S. arrivo > < h : m > < h : m >
(part) arrivo

- Leggo da tastiera { Staz. part. / Staz. arrivo / tempo (h) }
- Trovare i treni che partono dalle staz. definite e che hanno un tempo di percorrenza < di quello definito.
- Ordinarle in base all'ora di partenza e scriverle su un file d'uscita
- file d'ingresso e d'uscita sono passati da L.C.



Ti: 18:30:10 (5 4 3 2 1 0) tf: 19:30

a = atoi(Ti[5])
b = atoi(Ti[4])
c = atoi(Ti[2])
d = atoi(Ti[1])

$tinizz = (10*c + d) + (10*a + b) * 60$

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main (int argc, char *argv[]){

    char num[15], datamax[11], data[11], tipo;
    int i, durata, durtot=0, max=0;
    float minuti, costo;

    if (argc != 2){
        printf ("Errore l.c.\n");
        return -1;
    }

    FILE *fp;
    fp= fopen ("conto.dat", "r");
    if (fp == NULL){
        printf ("Errore apertura\n");
        return -2;
    }

    i=0;
    while (fscanf (fp, "%s %s %d %c", num, data, &durata, &tipo)!=EOF){
        if (strcmp (num, argv[1])==0 && tipo=='U'){
            durtot+=durata;
            if (durata > max){
                max = durata;
                strcpy (datamax, data);
            }
        }
        i++;
    }
    printf ("Righe lette = %d\n", i);
    minuti = (float) durtot /60;
    costo = minuti * 0.12 + 0.20;
    printf ("Minuti totali in uscita = %.2f\n", minuti);
    printf ("Durata chiamata piu' lunga = %.2f in data ", (float)max/60);
    puts (datamax);
    printf ("\n Costo totale = %.2f\n", costo);
    fclose (fp);
    return 0;
}
```

DOMANDE 1) 11101

- Se BIN PURO : $16+8+4+1 = (29)_{10}$.
- Se M&S : $-(8+4+1) = (-13)_{10}$
- Se CA2 : $-16+8+4+1 = (-3)_{10}$

2) IMMAGINE 320 x 200 pixel (256 colori) Memoria?

3) $F = C\cdot(A+\overline{B}) + \overline{B}\,\overline{C} + ABC$

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 75 di 120

```
# include <stdio.h>
# include <stdlib.h>
# define RMAX 20
int main (int argc, char *argv[]) {

    int R, i, J, giorno[RMAX], durata[RMAX], K, tot, control, flag;
    char attivita[30+1];

    if (argc != 2) {
        printf ("Errore e.c. \n");
        return -1;
    }

    FILE *fp;
    fp = fopen ("time.dat", "r");
    if (fp == NULL) {
        printf ("Errore \n");
        return -2;
    }
    i = 0;
    control = 1;
    while (fscanf (fp, "%d %s %d", &giorno[i], attivita, &durata[i]) != EOF) {
        if (giorno[i] < 0 || giorno[i] > 365)
            control = 0;
        i++;
    }
    R = i;
    printf ("Righe lette = %d \n", R);
    if (control == 0) {
        printf ("Errore formato giorni \n");
        return -3;
    }
    fclose (fp);

    K = atoi (argv[1]);
    flag = 0;
    for (i=0; i<R-1; i++) {
        tot = durata[i];
        for (J = i+1; J<R; J++) {
            if (giorno[i] == giorno[J])
                tot += durata[J];
        }
        if (tot > K) {
            flag = 1;
            printf ("Limite superato, giorno %d con totale %d \n", giorno[i], tot);
        }
    }
    if (flag == 0)
        printf ("Dati nel file corretti, limite non superato \n");

    return 0;
}
```

PROGRAMMAZIONE

conversione RLE di un file

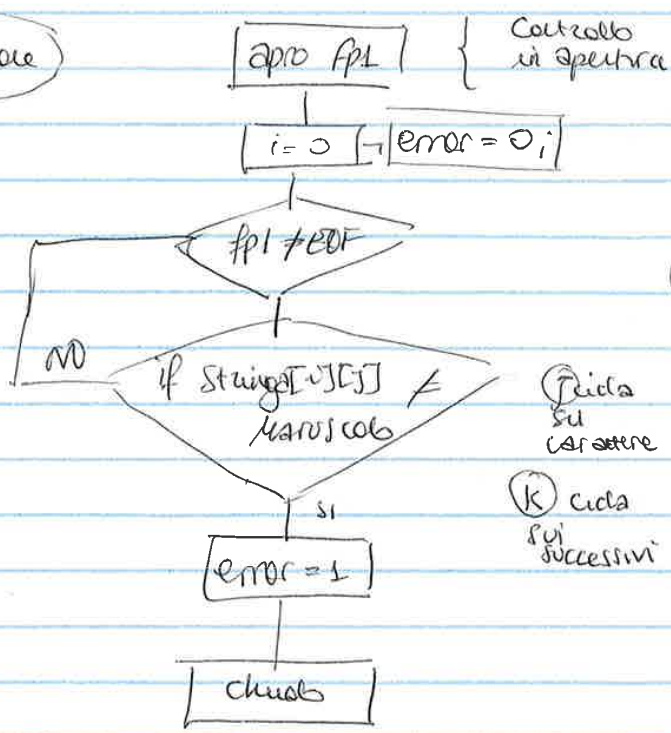
Es: AAAA BB CCC ⇒ A4 B2 C3

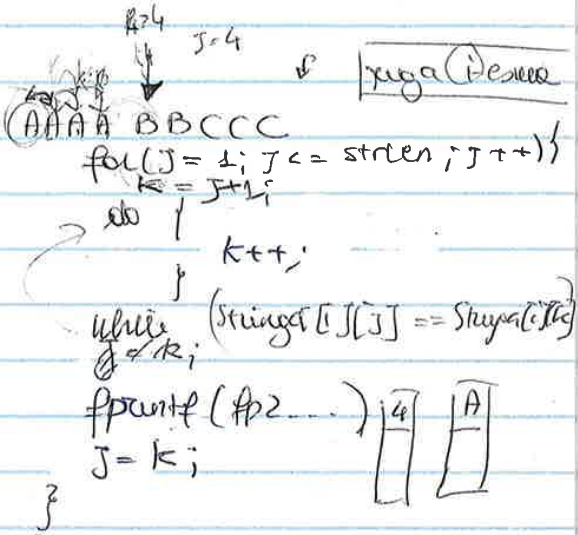
al max [512]

argc { · nome del sorgente
· nome file uscita

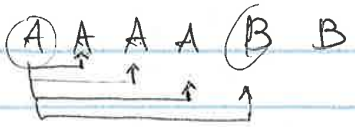
Stringa [i][512]

inizializzazione






```
while (__ = __ ){
    k++;
}
```



j = k

printf ("Caratteri file 1 = %d\n Caratteri file 2 = %d\n Compressione = %.2f\n") sum1, sum2, (float)(sum1-sum2)*100/sum1);
return 0;
}

0 1 2 .. 9 A B C .... F
(15)

30/01/2012

① $n_1 = (FA00)_{16} \Rightarrow$ 1111 1010 0000 0000

se CA2 = $-2^{15} + 2^{14} + 2^{13} + 2^{11} + 2^9$ =

$-32768 + 16384 + 8192 + 2048 + 512 =$

$= (-5632)$ CA2

se M&S = -27136

$n_2 = (8000)_{16} \Rightarrow$ 1000 0000 0000 0000

se CA2 = $-2^{15} = -32768$

se M&S = -0.

② TABELLA VERITA' $f = (a \oplus b) + c + (\bar{a}b\bar{c} + ab\bar{c})$

A	B	C	a⊕b	f
0	0	0	0	1
0	0	1	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	0	1
1	1	1	0	1

? 1

③ Codifica RASTER → l'immagine è rappresentata punto x punto (FOTOGR.)
Codifica VETTORIALE → " " " " con delle curve (DISEGNI)

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 81 di 120

```
# include <Stdio.h>
# include <Stdlib.h>
# include <String.h>
# define RR 24
# define CC 70
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define RR 50

int distinti (char nome[], char v[][30], int n);
int main (int argc, char *argv[]) {
        char anno[5], artista[RR][50], album[50], nome[30];
        int R, i, N=1;
        if (argc!=4) {
                printf ("Errore c.c.\n");
                return -1;
        }
        FILE *fp;
        fp = fopen ("album.txt", "r");
        if (fp == NULL) {
                printf ("Errore Apertura\n");
                return -2;
        }
        if (strcmp ("-a", argv[2])==0) {
                i=0;
                while (fscanf (fp, "%s %s %s", anno, artista[i], album) != EOF) {
                        if (strcmp (argv[3], anno)==0)
                                printf ("%s %s\n", artista[i], album);
                        i++;
                }
                N = distinti (nome, artista, N);
                printf ("Elementi distinti = %d\n", N);
        }
        else if (strcmp ("-b", argv[2])==0) {
                i=0;
                while (fscanf (fp, "%s %s %s", anno, artista[i], album) != EOF) {
                        if (strcmp (argv[3], artista[i])==0)
                                printf ("%s %s\n", anno, album);
                        i++;
                }
        }
        fclose (fp);
        return 0;
}
int distinti (char nome[], char v[][30], int n) {
        int i, flag=0;
        for (i=0; i<n && flag==0; i++) {
                if (strcmp (v[i], nome)==0)
                        flag=1;
        }
        if (flag==0) {
                strcpy (v[n], nome);
                n++;
        }
        return n;
}
```

Funzione

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define RR 20

int main (int argc, char *argv[]){
    char nome[30+1], cognome[30+1], naz[RR][4];
    int p[RR][10], i,j,k,n, sum, tot;
    FILE *fp;
    fp = fopen ("score.txt", "r");
    if (fp==NULL) {
            printf ("Errore Apertura \n");
            return -1;
        }
    if((argc==4) && (strcmp ("-a", argv[2])==0)) {
        i=0;
        while(fscanf (fp, "%s %s %s %d %d .... %d", nome, cognome, naz[i],
                        &p[i][0], &p[i][1], &p[i][2], .... &p[i][9])!=EOF){
            if (strcmp (naz[i], argv[3])==0) {
                sum=0;
                for (k=0; k<10; k++) {
                    sum = sum + p[i][k];
                }
                printf ("%s %s, penalita'= %d\n", nome, cognome, sum);
            }
        i++;
        }
        fclose(fp);
    }
    if ((argc == 4) && (strcmp ("-b", argv[2])==0)){
        n= atoi (argv[3]);
        i=0;
        while (fscanf (fp, "%s %s %s %d %d .... %d", nome, cognome, naz[i], &p[i][0]
                        &p[i][1], &p[i][2], .... &p[i][9]) != EOF) {
            if (p[i][n-1] == 0) {
                printf ("%s %s, %s\n", nome, cognome, naz[i]);
            }
        i++;
        }
        fclose (fp);
    }
    if((argc ==3) && (strcmp ("-c", argv[2])==0)) {
        i=0;
        tot=0;
        while (fscanf (fp," %s ..... // come prima // )!=EOF){
            for (k=0;k<10;k++)
                tot = tot + p[i][k];
            for (j=i+1; j<R; j++) {
                if (strcmp (naz[i], naz[j])==0) {
                    sum=0;
                    for (k=0; k<10; k++)
                        sum = sum + p[j][k];
                    tot = tot + sum;
                }
            }
            printf ("%s , totale = %d\n", naz[i], tot);
        i++;
        }
        fclose (fp);
    }
    else {
        printf ("Errore L.C.\n");
        return -1;
    }
    return 0; }
```

la parte col 3° flag non gira

```
# include <stdio.h>
# include <stdlib.h>
# include <string.h>
# define RR 14
int main (int argc, char *argv[]) {
    struct giocatore {
            char nome [30+1];
            char cognome [30+1];
            char ruolo [20+1];
            float km;
            int g[RR], pass, tiri, ff, fs;
        };
    struct giocatore p;
    int i, max;
    FILE *fp;
    fp = fopen ("stat.txt", "r");
    if (fp == NULL) {
            printf ("Errore Apertura\n");
            return -1;
        }
    if (argc == 4 && strcmp ("-a", argv[2])==0) {
            i=0;
            while (fscanf (fp, "%s %s %s %f %d %d %d %d %d",
                    p.nome, p.cognome, p.ruolo, &p.km, &p.pass, &p.tiri, &p.g[i], &p.ff, &p.fs)!=EOF) {
                if (strcmp (argv[3], p.ruolo)==0)
                printf ("%s %s: km %.1f, passaggi %d, tiri %d, goal %d, falli fatti %d, falli subiti %d\n",
                    p.nome, p.cognome, p.km, p.pass, p.tiri, p.g[i], p.ff, p.fs);
                i++;
            }
            fclose (fp);
        }

    if (argc==4 && strcmp ("-b", argv[2])==0) {
        i=0;
        while (fscanf (fp, "%s %s ..... %d", p.nome, p.cognome, ..., &p.fs)!=EOF) {
            if (strcmp (argv[3], p.ruolo)==0)
                printf ("%s %s : km %.1f .... //come prima// ",
                    p.nome, p.cognome, p.km ...);
            i++;
        }
        fclose (fp);

    if (argc==3 && strcmp ("-c", argv[2])==0) {
        i=0;
        max=0;
        while (fscanf (fp, " ... // come prima// ) !=EOF) {
            if (p.g[i] > max)
                max = p.g[i];
            i++;
        }
        fclose (fp);
        fp = fopen ("stat.txt", "r");
        i=0;
        while (fscanf (fp, " // come prima // ) != EOF) {
            if (p.g[i] == max)
                printf ("%s %s, %s, goal = %d\n", p.nome, p.cognome, p.ruolo, max);
```

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 89 di 120

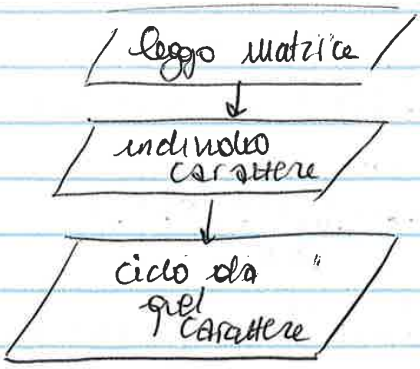
PROGRAMMAZIONE

R = 6
C = 7

L.c. (prg.exe forza4.txt -a/ -b giallo/rosso) argc = 4
argv[1] argv[2] argv[3]

2) flag = "-a".

(da controllare)




```
if ( ---------- ) {
    trovato = 0;
    for (k=j; k<4; k++){
        if m[i][j] != m[i][k]
            trovato = 1;
    }
    if (trovato==0){
        printf (forza4 orizz)
```

(x3)

orizz. verticale, diagp.

b) flag "-b".

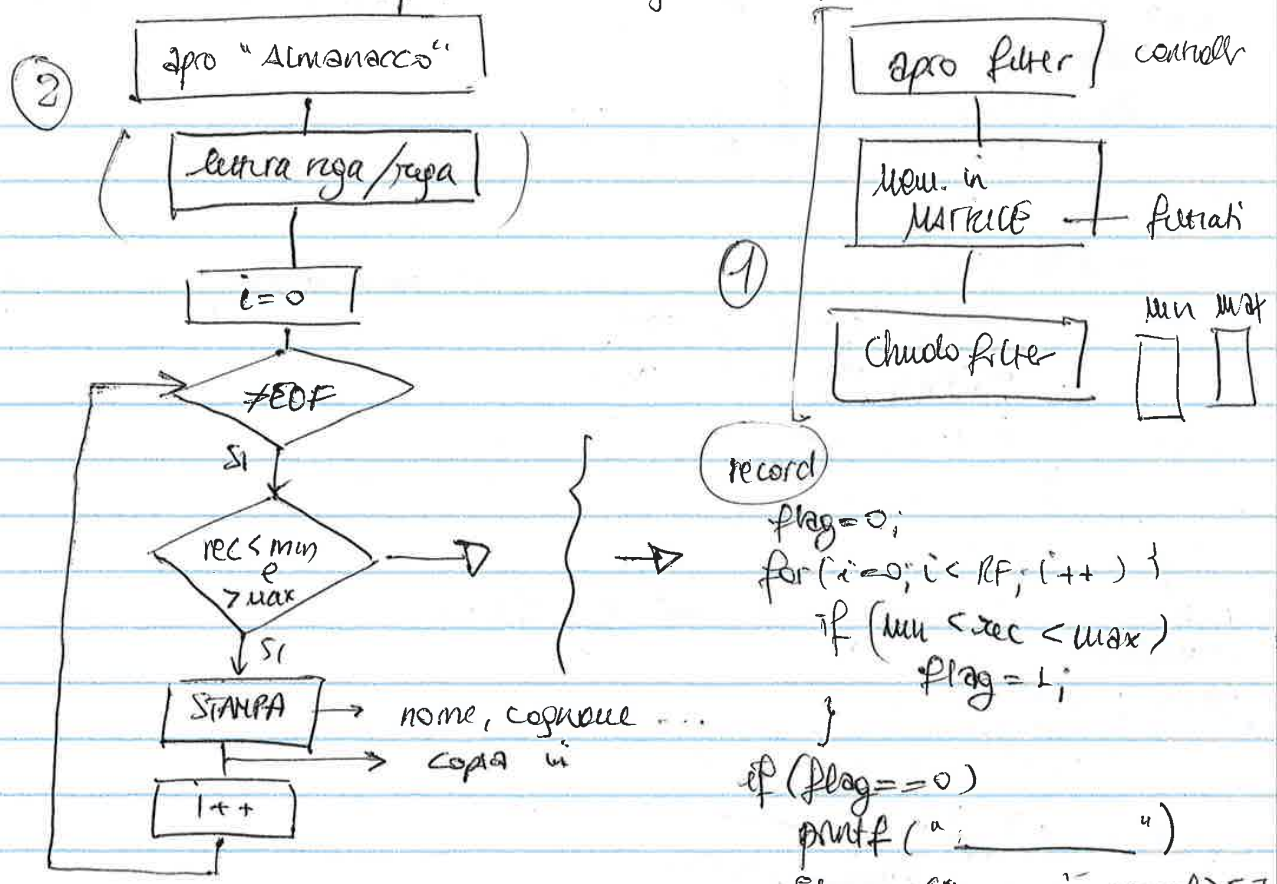
```
if (      ) {
    orizz = 0;
    for ( k = j; k < 3; k++) {
        if (m[i][j] != m[i][k])
            orizz = 1
    }
    if (orizz = 0) {
        if ( m[i][j+3] == '0')
            printf ("trovato orizz, sposta nella colonna j+3")
```

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define R 6
#define C 7
int main ( int argc, char *argv[]) {
    int orizz, vert, diagp1, diagp2, i, j, k;
    char m[R][C], c;
    FILE *fp;
    fp = fopen ("forza4.txt", "r");
    if (fp == NULL) {
        printf ("Errore Apertura\n");
        return -1;
    }
    for (i=0; i<R; i++) {
        for ( j=0; j<=C; j++) {
            fscanf ( fp, "%c", &m[i][j]);
    }   }
    fclose (fp);
    if (strcmp ("-a", argv[2])==0 && (strcmp ("rosso", argv[3])==0 ||
        strcmp ("giallo", argv[3])))
```

```
vert = 0;
for (k=i; k<i+3; k++) {
    if (m[k][j] != c)
        vert = 1;
}
if (vert==0) {
    if (m[i-1][j] == '0') || (m[i+3][j] == '0'))
        printf ("Sposta nella colonna %d\n", j+1);
}
diag1 = 0;
for (k=i; k<i+3; k++){
    for (h=j; h<j+3; h++) {
        if (m[k][k] != c)
            diag1 = 1;
    }
}
if (diag1 == 0) {
    if (m[i-1][j-1] == '0')
        printf ("Sposta nella colonna %d\n", j);
    else if (m[i+3][j+3] == '0')
        printf ("Sposta nella colonna %d\n", j+4);
}
diag2 = 0;
for (k=i; k      ; k--) {
    for (h=j; h      ; h--) {
        if (m[k][k] != c)
            diag2 = 1;
    }
}
if (diag2 == 0) {
    if (m[i-1][j+1] == '0')
        printf ("Sposta in colonna %d\n", j+2);
    else if (m[i+1][j-1] == '0')
        printf ("Sposta in colonna %d\n", j);
}
}
}
}
}
else {
    printf ("Errore linea di comando\n");
    return -2;
}
return 0;
}
```

Algoritmo

. Elenco dei cadetti che rispettano le regole del filtraggio.




```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define RR 100

int main (int argc, char* argv[]){
    char nome[40+1], cognome[40+1], data[10+1],
         nomef[RR][40+1], cognomef[RR][40+1];
         dataf[RR][40+1];
    int RF, i, j, k, p, pf[RR], flag, pmax=0, pmin=1000, posmax, posmin, R, a[RR],
        m[RR], g[RR], amax;
    float record, recf[RR], min[RR],
          max[RR];
    if (argc!=2) {
        printf("Errore N.C.\n");
        return -1;
    }
    FILE *fp1, *fp2;
    fp2= fopen("filter.txt", "r");
    if (fp2==NULL){
        printf("Errore apertura filter\n");
        return-2;
    }
    i=0;
    while(fscanf(fp2, "%f - %f", &min[i], &max[i])!= EOF) {
        i++;
    }
    RF=i;          // Regole filtraggio //
    fclose(fp2);

    fp1= fopen("almanacco.txt", "r");
    if(fp1==NULL){
        printf("Errore apertura almanacco\n");
        return-3;
```

19/07/2013 (B)

Teoria ①

$(1111\ 0001)_{MS} \rightarrow -(2^6+2^5+2^4+1) = (-113)_{10}$

$(110001)_{MS} \rightarrow -(2^4+1) = (-17)_{10}$

$(0001)_{MS} \Rightarrow (+1)_{10}$

② Tabella di verità:

f sempre 0 se $f = a(b+c)\bar{a}$

($a\bar{a}$ da sempre zero)

③ La codifica ASCII permette di rappresentare qualunque carattere alfanumerico in una sequenza binaria di 8 bit.

Programmazione

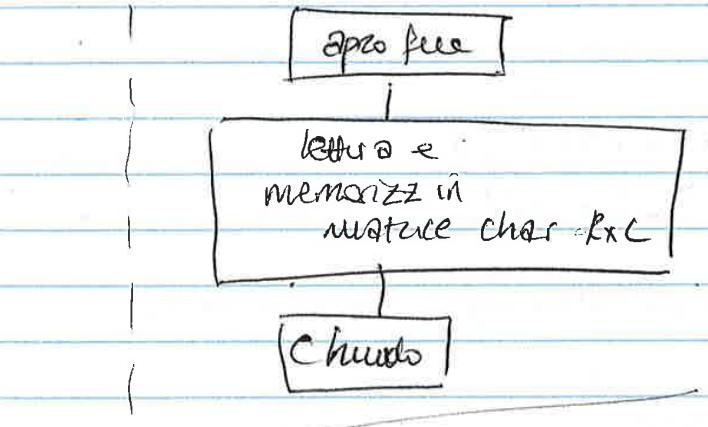
RC

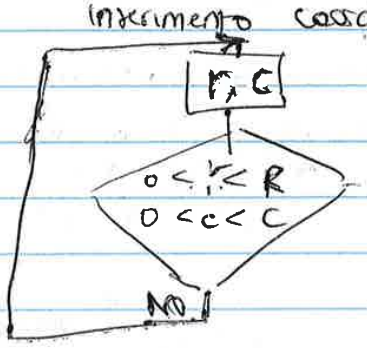
L.C. | Prog.exe / mappa.dat

argc = 2 ⇒ argv[1]

\# muro
(spazio) isola

• Inserimento di coordinate fino a (-1, -1)






```
do
{ printf("Coordinate non valide");
  scanf("%d%d");
}
while( r<0 || r>R || c<0 || c>C );
```

```
while( r!=-1 && c!=-1)
{
// inserisci coordinate //
if(r<0 ...)
printf("non valide
ripetere")
```

Se trovo # devo cercare a destra, sinistra in alto e in basso

```
for(h=x+1;h<R;h++)
  if(m[h][c]=='#')
    conta++;
```

Isola: conta+1 conta+1

```
else
for(h=r-1;
h>0;h--)
if( )
conta++;
for for...
```