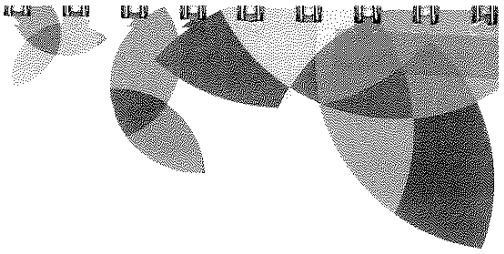
NUMERO: 838 DATA: 25/02/2014

A P P U N T I

STUDENTE: Chirico

MATERIA: Informatica + Eserc.

Prof. Mezzalama



INFORMATICA

6 unità logiche astratte
1) Unità input: ingresso dati
2) Unità output: uscita dati
3) Unità di memoria primaria: ad accesso rapido, contiene input
4) Unità aritmetica e logica ALU: esegue calcoli e decisioni logiche
5) Unità di elaborazione centrale: CPU: sede amministrativa coordina operazioni
6) Unità di memoria secondaria: magazzino a lungo termine

all'inizio: elaborazione batch: 1 processo per volta, poi multiprogrammazione, + processi assieme.
elaborazione personale → ognuno lavora sua macchina
elaborazione distribuita → lavoro fatto da rete di computer
client/server → server mette a disposizione per + client programmi e info.

linguaggi:
- macchina: immediatamente comprensibile da pc, dipende da macchina
- assembly: abbreviazioni per operazioni elementari
- alto livello: per accelerare programmazione: + operazioni in 1 istruzione, tradotti in ling mac da compilatori.

linguaggi:

BCPL (1967, Richards) + B (Thomson, 1970)

C è loro evoluzione sviluppata da Dennis Ritchie, 1972
usato per scrivere sistemi operativi di tutti i computer
→ standardizzato e approvato nel 1989, per fare sì che girasse su tutti i computer

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 3 di 116

\n → andare alla riga successiva (newline)
\t → andare alla tabulaz. successiva
\r → inizio riga corrente
\\ ; \' ; \" → visualizza backslash, apice, virgolette
Linker = insieme funzioni, programma predotto dal linker

printf("Welcome\n To\n Turin");
→ printa a video a parola x riga:

```
Welcome
To
Turin
```

#include <stdio.h> → per preprocessore, includere file di intestazione per ci/o standard, per trovare + facilmente errori

int, integer2, integer2, sum; → dichiarazione, nomi delle variabili, di tipo int, interi

NB: è case sensitive!! integer → identificatore, non + di 31 caratteri

dichiarazioni vanno prima di istruzioni
errore di sintassi → compilatore non riconosce un'istruzione
printf("Enter 2° integer\n") →

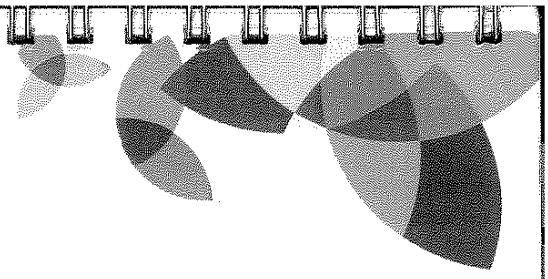
scanf("%d", &integer1); → funz. scanf prende dati da tastiera

2 argomenti

%d → stringa di controllo del formato: intero decimal
in
simile funzione del carattere di escape
&integer → indica dove immagazzinare la variabile
in
operatore di indirizzo.

Sum = integer1 + integer2; → istruzione di assegnamento valore a variabile sum
in
operatore di assegnamento
+ → operatore binario.

Variabile = calcolo
 sx dx



↳ molto + piccola RAM

Unità di ingresso / Uscita

↳ permettono al PC di interagire col mondo esterno
- * leggibili da uomo: schermo, tastiera
- * leggibili da macchina: dischetti, cd.

ALU

↳ esegue operaz. aritmetico-logiche, integrata nel processore

Unità di controllo

preleva istruzione
la decodifica
la esegue

Caratteristiche

variabile → locazione con nome + tipo + valore
nuova assegnazione di a fa leggere valori precedenti

% → operazione di modulo, resto fra divisione fra interi
elevamento a potenza: pow

C → azioni
↳ prende decisioni
struttura di controllo "if"

operatori di uguaglianza - priorità op. relaz.
!= diverso == uguale.

operatori relazionali No spazi fra simboli!
> < >= <=

NB: associa a dx

NB: == : op. uguaglianza = : op. assegnamento

Priorità:

() * / % + - < > <= >= == != = → associano da sx a dx

da dx a sx

NB: nome variabile + parole chiave

NB: sono operatori binari

Strutture modificate: -Uper + casi:

```
if (grade >= 90)
    printf ("A\n");
else
    if (grade >= 80)
        printf ("B\n");
    else
        if (grade >= 70)
```

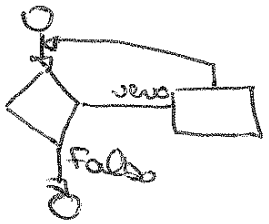
NB: se + istruzioni dopo if uso graffe:

```
if (condiz.) ——▷ istruzione composta
    {blocco};
```

While istruzione reiterata fino a che condizione rimane vera.

▷ ciclo: NB, deve dare condizione che prima o poi diventerà falsa!

```
product = 2;
while (product <= 100)
    product = 2 * product;
```



Totale ▷ variabile x accumulare somma serie valori
contatore ▷ var. per contare

▷ NB: devono essere prima azzerate !!

Valore sentinella: viene in input, dice quando terminare ciclo while

▷ iterazioni indefinite.

Processo top-down per sviluppo algoritmi

Def. la media della classe Top

↓

Inizializ. variabili
Prendere input e sommare valori
calcola e visualizza media

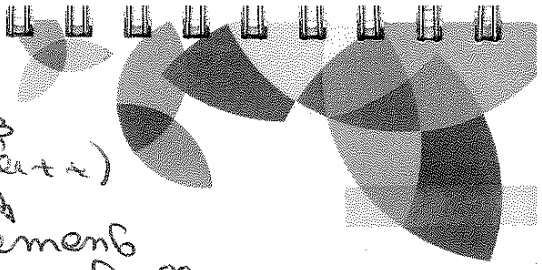
↓

raffinamenti successivi

↓

down

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 9 di 116



Iterazione for

exp 1 exp 2 exp 3

```
for (counter = 1; counter <= 10; counter++)
```

keyword + nome var. con val. iniziale → condiz. val. finale var. controllo → incremento var. controllo.

Equivalente a:

```
express 1;
while (express 2) {
    istruzioni;
    express 3; }
```

Si può avere sia incremento che decremento.

```
for (i = 7; i <= 77; i += 7)
```

↳ passi di 7 fino a 77

```
for (n = 2; n <= 100; sum += n * n += 2)
```

↳ indice + incremento con virgola

```
x ^ y = pow(x, y)
```

↳ #include <stdio.h>
#include <math.h>

double → variabile + grande di float.

18,673 → %.2f → 18,67

Switch

↳ selezione multipla, + scelte (case), caso opzionale (default)

```
switch (grade)        // variabile
{
    case 'A': case 'a':
        ++aCount;
        break.   -> per evitare che si eseguano tutti gli altri case
    .... altri case
```

→ espressione di controllo

```
while (grade = getchar())
```

↳ prende in input carattere e lo immagazzina in grade.

%c → carattere.

EOF: end of file, combinaz. tasti che termina sistema.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 11 di 116

Break: uscita dalla struttura, alla riga successiva

Continue: fa eseguire successiva iterazione del ciclo.

Operatori logici (binari)

&& And vera se entrambe espressioni vere

! Not (unario)

|| Or vera se una delle due è vera

&& + priorità ||

if (grade == -1)
printf("grade: %f\n", grade);

== : uguaglianza = : assegnamento.

Capitolo 6 VETTORI

vettori: strutture dati dello stesso tipo.
↳ + locaz. di memoria con stesso nome

v[1]
↳ 2° elemento
↳ nome vettore

NB: indice vettori inizia da zero, 6° elemento: v[5]

dichiarazione:

int c[12] = {0, 1, 3, 7, 12, 5, 9, 8, 13, 15, 32, 6};
↳ 12 elementi per vettore di interi

se non do valori, tutti elementi = 0 se azzero 1° elemento.

define SIZE 10
↳ uso tutte maiuscole per costanti simboliche.

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 13 di 116

printf("media è %d ", (float) ABC / SIZE);

- mediana:

~~for(i=0; i<=SIZE-2; i++)~~

```
if (a[i] > a[i+1]) {
    help = a[i];
    a[i] = a[i+1];
    a[i+1] = help; }
printf("mediana è %d", answer[SIZE/2]);
```

- moda

```
for(i=0; i<=size-1; i++)
    switch(answer[i]
        case '1'
            1counter++;
            break;
            ... (per altri casi)
for(i=0; i<=10
    if 1counter > (i+1)counter
        max = icounter;
    else
        max = (i+1)counter;
printf("moda è %d", &i);
```

Ricerca nei vettori

ricerca lineare → confronta ogni elemento con chiave di ricerca (inefficiente)

```
for(i=0; i<size; i++)
    if (searchkey == a[i])
        printf("searchkey è uguale a: a[%d] ", &i);
        break; }
```

ricerca binaria → vedi ordinati, confronto con elem. di mezzo, se è >, scarto tutti gli elem. a sx se è <, scarto elem. a dx, se è = l'ho trovato

Funzioni c.Type:
- isdigit(int c) √se cifra
- is_alpha(int c) √ se caratt. alfab
- isalnum(int c) √ se cifra o lettera
- islower(int c) √ se minuscolo
- isupper(int c) √ se maiusc.
- isspace(int c) √ se spazio
- ispunct(int c) √ se punteggiatura

tolower → converte maiusc. in minusc.
toupper → converte minusc. in maiusc.

Cap 7: I puntatori

puntatori: variabili che come valore hanno indirizzi di memoria riferimento indiretto a variabile.

int *countPtr; *nome → Ptr. punta a un oggetto intero,
↳ operatore di deriferimento memorizzato in countPtr.

Inizializzazione:

```
int y = 5;
int *yPtr;
yPtr = &y;
```

↳ operatore di indirizzo, assegna a yPtr l'indirizzo di y

printf("%d", *yPtr); → visualizza il valore di y

per vedere indirizzo:

```
printf("The address of y is: %d \n", &a);
```

↳ è lo stesso del valore di *yPtr, avrei potuto scrivere yPtr al posto di &a

```
printf("%p", &a);
```

↳ carpet locazione memoria (in esadecimale)

```
printf("%p", aPtr);
```

Memorizzazione

k → $10^3 \sim 2^{10}$ (ga 1024, ~ 1000)
M → $10^6 \sim 2^{20}$
G → $10^9 \sim 2^{30}$
T → $10^{12} \sim 2^{40}$
P → $10^{15} \sim 2^{50}$

1 pagina → 2000 Byte
1 giem → 1 GB

: bit : per second NB: uno bit e non byte

Pascalina (pascal, 1642) → calcolatrice
1° software per comandare strumento:
Jacquard, 1801, tessitura, macchine per tessere
Babbage → primo informatico
Eckert-Mauchly → 1° calcolatore elettronico
1980 → PC, low cost computing.
1980 → memorizzazione dati
2000 → comunicazione
2005 → web.

Calcolatori
- → general: pc, workstation, server, Mainframe, supercomputer.
- → dedicati: (1 solo programma) cellulari, MP3, auto

Lezione 2 (Linguaggi per elaboratori)

Programmare
- → Problema
- → soluzione
- → Soluzione formale
- → realizzazione

→ sviluppo algoritmo: descrizione formale sequenza azioni che devono essere eseguite per risolvere problema in tempo finito.

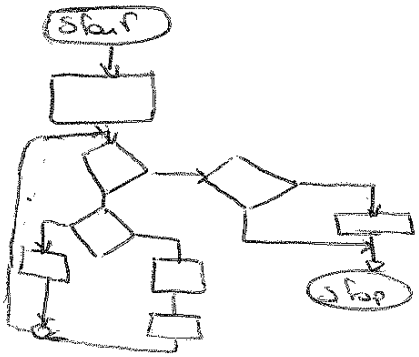
deriva da un arabo del 700 d.C. matematico che scrisse libro algebra

max tra A e B

m = 0
se A > B m = A altrimenti
se B > A m = B altrimenti.

} soluzione formale

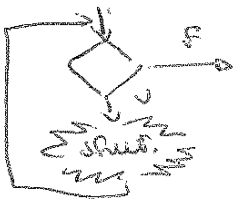
Lezione 3 (Diagrammi di flusso)



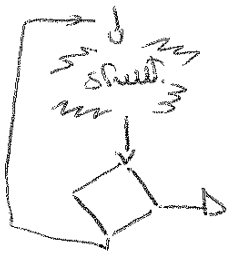
- if - Then - Else

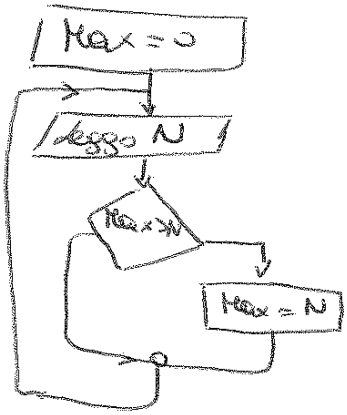
blocco condizionale

- While - Do



Do while:





struttura iterativa per trovare max tra n numeri
utilizzo variabile:
ciclo = ciclo + 1
deve essere uguale a n, volte che si ripete

Lezione 4 (Architettura degli elaboratori)

- Unità di ingresso: (di input dati)
  - tastiera
  - sensori

  da alfab. est. a l. macchina
- Unità di uscita
  - video
- Memoria (programmi + dati) → RAM (centrale) di massa
- Unità di elaborazione (collegata a RAM)
  - CPU (microprocessore)
    - → unità decodifica istruzioni
    - → unità gestione indirizzi *
    - → ALU (Aritmetica-logica Unit)
    - → unità gestione BUS *
    - → registro (memoria locale con dati utili in quel momento accessibile velocemente).

* gestione dati

memoria esterna (permanente)
- Tempo di accesso
  Ta=k ~ 60 ms.
  su un disco:
  Ta ~ ms.

RAM → Programmi + sist. operativo + memoria video (Ram video)
ROM → Progr. d'avvio,
  Bios: Basic input output system

RAM → SRAM
  → DRAM (DDR2 o DDR3)

EEPROM → Tipo di ROM, tecnologia base delle flash memory, scrivere e cancellare con facilità

all'inizio VCPU = VRAM, poi VCPU > VRAM
1 ns 60 ns

trasferisco in CACHE dati + di frequente, 8 Mb, sono molto + costose
→ organizzata su + livelli

Multi-core
CPU + processori unico pacchetto, 1 processore → + CPU

Lezione 5 (Introduzione al C)

linguaggio ad alto livello
↓
compilatore
↓
ling. macchina

- Fortran
- Cobol
- basic ling. violetto
- C (Java, PHP, Python)

linguaggio → di alto livello
  → simbolici
  → di macchina

- Parole chiave
  Dep. a priori
- istruzioni

Identificatori
nomi dati

main() → dichiara i var globale
{ dichiarazioni locali
parte esecutiva → I modulo
}

Costanti:
const <tipo> <costante> = <valore> (all'interno di un modulo)
↳ spazio di solo lettura
const char c = 'b' NB: virgolette

visibilità variabili
↳ in ogni modulo (globali)
↳ in solo modulo (locale)

printf(" "); → visualizza messaggio

variabile controllo
↳ for (inizializza, valida, aggiorna)

printf("frase %d", nome variabile)
↳ voglio stampare variabile intera
%f → reale %c → carattere

S = a + b
var. espressione

Lezione 6 (Numeri binari)

Codifica informazione
ogni dato è rappresentato da 0 e 1
n bit → 2^n combinazioni binarie diverse.
per codificare k oggetti:
↳ $n >= \log_2 k$

numerazione posizionale:
$A = \sum_{i=0}^{N-1} a_i B^i$ in base B, ha B cifre: {0 ... B-1}

321 base 4: $3 \cdot 4^2 + 2 \cdot 4 + 1 \cdot 4^0$
$4^2\ 4\ 4^0$
base 10

1 e 0 → interruttore chiuso o aperto → binario
(hardware)
base 2 → 2 posizioni

-5 Rappresentazione in complemento a 2

1) +5 → 0 1 0 1
2) inverti bit a bit 1 0 1 0
3) somma 1 1 0 1 1

```
11011   -5
00100   +6
-------
00001
```

Lezione 7 (Codifica numeri reali)

Rappresentazione reali

↳ valore + posizione virgola
↳ floating point

0 1 1 . 1 1
$2^2\ 2^1\ 2^0\ 2^{-1}\ 2^{-2}$ potenze di 2^{-k} $k>0$ virgola fissa

floating point → + ampio intervallo di rappresentazione.

$\pm$ Mantissa $\cdot 2^{x}$ (x: base della rappresentazione)

3 elementi: segno, mantissa, esponente

Java → floating point

1 bit → segno
8 bit → espon.
23 bit → mantissa } IEEE 754 SP:

1 bit → segno
11 bit → esp.
52 bit → mant. } IEEE 754 DP:

Es:

1 111 0101
s esp mant.

111 → complemento a due

$-\underbrace{0.101}_{\text{n° assoluto}} \cdot 2^{\underbrace{111}_{\text{n° relativo}}} \rightarrow -5 \cdot 2^{-1} = -2.5$

$-2^2 + 2^0 + 2^{-1}$

-27

+27 inversione bit a bit (complementazione)

+1

11100

00011 → positivo

+1

00100 → +4 → quindi è -4

esadecimale 8 bit

3A

0011 1010

segno modulo

\+

mod & segno:

A → 10 8+2

$2 + 2^5 + 2^4 + 2^3$

2+16+8+32 = 58

Lezione 8 (Input da C, variabili ed espressioni)

istruzioni elementari

- assegnazione
- input/output

Assegnazione

<variabile> = <valore> <variabile> = <espressione>

Programma:

1) dichiarazione variabili (tipo e valore)

dichiarazione

<tipo> <nome> (in complemento a due int)

double: floating point → 64 bit

float → 32 bit

posso anche inizializzare in fase dichiarat:

int a = 3;

- printf() → output
- scanf() → input

} #include <stdio.h>

operatori di confronto

== > >= } output booleano, V o F
!= < <=

Lezione 09 (Semplici esercizi)

op. booleani AND, OR, NOT connettono op. confronto

```
#include <stdio.h>
#include <stdlib.h>
int main()
{ }
```
} scheletro fondamentale

casting: cambia tipo di espressione, risolve problema divisione interi

(int) g; → riduce g in int
(float) b*h/2

Lezione 10 (Logica Booleana)

operatori di incremento:

a++ aggiungo 1 ad a
a-- sottraggo 1 ad a

posso applicarlo:
- a++ notaz. postfissa
- ++a notaz. prefissa

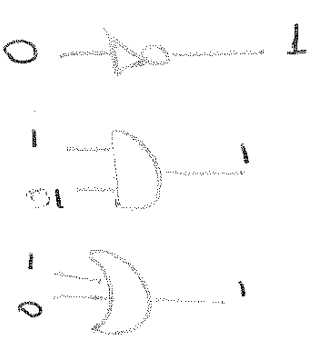
a+=2 : a = a+2

Operatori logici

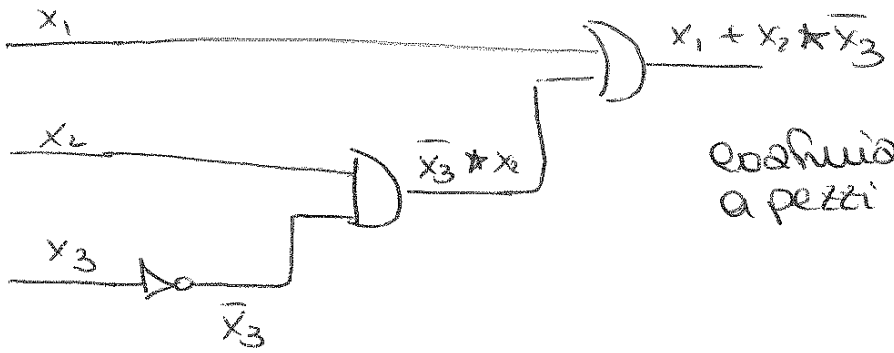
!	&&	||	si applicano ad operatori di confronto
NOT	AND	OR	

↳ inversione logica
(x > 0) → può essere V o F, è espressione logica
1847 George Boole, introduce nuovo tipo di logica formale
solo due valori: V o F
frasi = + affermazioni = + variabili booleane
- OR → V se V almeno 1 dei due

Le operazioni hanno corrispondente circuito base chip



Es: $x_1 + x_2 * \overline{x_3}$ circuito:



costruisco circuito a pezzi

Combinazione di circuiti di base

Compare NOT, AND, OR.

↳ posso anche fare sommatori, moltiplicatori, cabine comparatori

possono rappresentare condizioni complesse

Lezione 11 (istruzione If)

caar: `g=(double)10;`

`(tipo) <espressione>;`

Cambia tipo della variabile.

Sizeof(int): dà n° byte di quel tipo

Istruzione if

boolean

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

se blocco = + istruzioni delimito con parentesi graffe

posso fare nelle annidate, l'una dentro l'altra

Lezione 12: (Esempi if e codifica immagini)

Es: stesso blocco listing. net:

if ($l_1^2 = l_2^2 + l_3^2$ || $l_2^2 = l_3^2 + l_1^2$ || $l_3^2 = l_2^2 + l_1^2$.

NB. $l_1^2 = l_1 * l_1$

if(x >= 2 && x <= 72) uso op. booleani per evitare if annidati

Informazione multimediale

suoni
immagini

Immagini

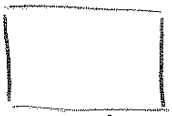
↳ info geometrica
↳ info singoli bit, + elementi, pixel, matrice di pixel

bit map:

n° bit coerente con n° pixel memoria

1 pixel → RGB

24 bpp (true color → 8R 8G 8B
16 bpp → 5R 6G 5B



pixel → 3 byte

n° pixel × n° byte/pixel = $\frac{10000}{3}$ byte

Codifica vettoriale autocad

↳ rappresenta una geometria

file vettoriale → file bitmap (raster) → schiaggio pixel

lavoro scheda grafica

Compressione dati

↳ lossless permette recupero dati iniziali
• zip → file di testo
↳ lossy non consente dati iniziali
jpeg

* run length

AAABBAA

A4 B2 A2 → contenuto

* dizionario

i Torinesi abitano a Torino

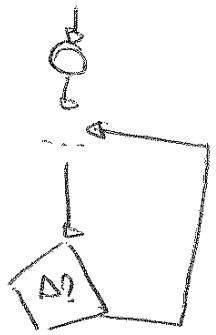
à = Torin (stringhe di caratteri)

CELP → Code Excited Linear Prediction, based on... usato per telefonia

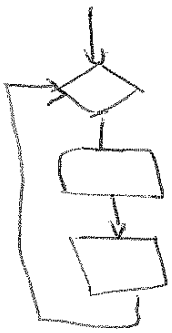
• cast: cambio tipo variabile
  fa (double) 10

• sizeof (tipo)

Cicli: "Fai questo fino a che..."
Ripetizione controllata da condiz. booleana.



oppure:



NB: attenzione ripetizione ∞

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

```
while(a>=10)      a=0
  {a++;}          ciclo si ripete 11 volte
```

possiamo contare per controllare cicli

```
i=0
som=0
while(i<10)                    → somma 10 numeri
  {i++;
   scanf("%d",&a);
   som+=a;
  }
```

Lezione 14 (cicli while esercizi)

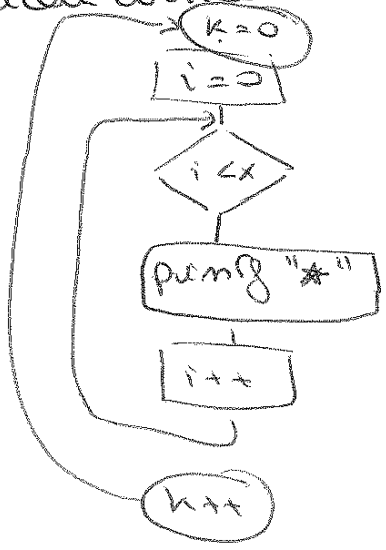
problema: verifica n° infrodotto da tastiera
fine n° info è zero

```
scanf("%d",&n);
while(n!=0){
    i++;
    if(n%N==0)
        printf("divisibile\n");
```

```
while(m2!=0) {
        diff=(m2-m1);
        if (diff > dmax) {
              dmax=diff;
              m1_m = m1;
              m2_m = m2;
            }
        m1=m2;
        printf("m2 =");
        scanf("%d", &m2);
        }
printf(" diff = %d\n", dmax);
printf("%d %d ", &m1, &m2);
```

Disegno 4 righe
└ Disegno X *

Ciclo while annidato




```
k=0
while(k<y) {
    i=0;
    while (i<x) {
    printf("*");
```

```
ris = n1 * n2;
break;
case '+':
    ris = n1 + n2;
    break;
case '-':
    ris = n1 - n2;
    break;
case '\':
    ris = n1 \ n2;
    break;
default
    printf("l'operatore inserito non è valido");

return 0;
    → per ritornare ad ambiente di lavoro iniziale
```

```
operazione ristorante
Switch (piatto) {
    case 1
        BR += 7.5 * quant;      /* primo */
        break;
    case 2
        BR += 7.5 * quant;      /* secondo */
        break;
    case 3
        BR += 5 * quant;        /* dolce */
        break;
    default
        printf("non valido")
```

per + piatti → inserisco switch all'interno ciclo while.

NB: attenzione in cui si deinizializzano variabili, qui: BR = 0.

```
while (quant != 0 || piatto != 0)
```

```
for(k=0; k<(n-i*2); k++)
        {printf("*")}
printf("\n");
}
```

Lezione 18 (Vettori)

Continue: non fa niente, riprendi da capo il ciclo

Vettore (struttura dati):
finora dati come entità singole
vettore → struttura dati costituita dal dato n° di elementi ordinati e dello stesso tipo
è un'unica entità, alla dichiarazione configure

vet → tutto vettore
vet[3] → elemento del vettore in posizione 3
▷ posso sommare/sottrare elementi

Dichiarazione:
`<tipo> <nomevettore>[<dimensione>];`
se voglio accedere a un elemento:
`<nome vettore>[<posizione>]` NB: V[0]
per prendere in input elementi:

```
for(i=0; i<N; i++)      /*N n°elem vet*/
  {scanf("%d", &vet[i]);}
```

Lezione 19 (Es. sui vettori)

▷ Conviene: int vet[N], così se cambio lung. vettore cambio solo valore della costante N all'inizio

int vet[3]={1,2,-7} → inizializzare vettore

vet[100]={0} → se metto un solo elemento lui assegna valore a tutti elementi

NB: "{}" funzione [illegible]

Es: verifico se un numero ∈ V[N]
{ int V[N]={1,2,3,4,5};

```
int A[10];
srand(time(NULL));
r=rand;
for(i=0; i<10; i++)
    {A[i]=rand()%.11;}
```

genero vett. casuale con n° da 1 a 10

- n° elem < soglia

```
for(i=0; i<N; i++)
    {if(A[i]<S)
        count++;}
```

VC → tabulazione imprimf

Vett → Array variabile

1° metodo che

vecmin → Funzione che cerca il min e dem.

↳ lo ripeto in loop N-1 volte

N.B. calo numero di partenza i

Lezione 21 (esercizi vettori)

Algoritmo ordinamento

selection sort

```
for(j=0; j<DIM-1; j++)
    minind=j;
    for(i=j+1; i<DIM; i++)
        {if(A[i]<A[minind])
            {minind=i;}
    temp=A[minind];
    A[minind]=A[j];
    A[j]=temp;
}
```

ci mette tante operazioni → molto lento

ES: leggi di B[] e A[]? stampa 2° colla iniziale

```
int A[dimA], B[dimB]; < srand(TIME(NULL));
for(i=0; i<dimB; i++)
    B[i]=rand()%.11 → faccio lo stesso per A
```

```
int cubo(int a);   → dichiarazione e prototipo

int main()
{ ... }

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

devono essere uguali

corpo funzione

→ nome variabile da deve essere uguale

`<output type> nome( <tipo><nome var1>, <tipo><nome var2> ...)`

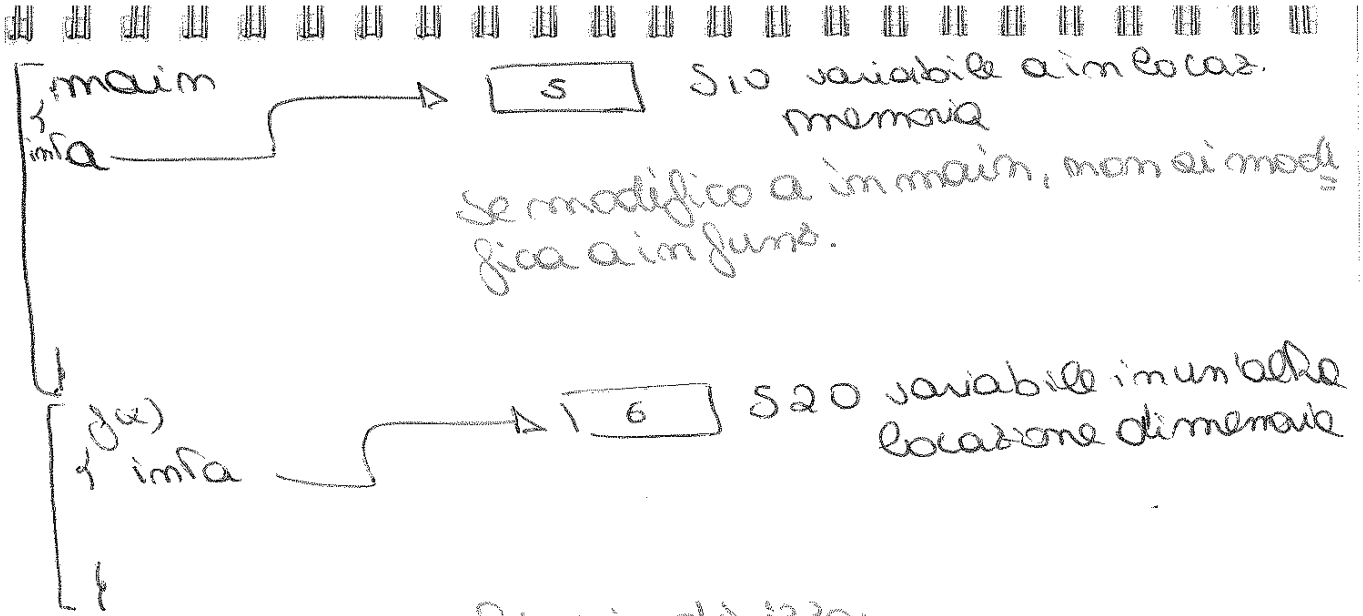
↳ da unico output

Void: non ritorna nulla, modificano qualcosa

NB: nel corpo non devo def. di nuove variabili di input

ES: leggo n1, e n2, output $n_1^{n_2}$

```
int exp(int x, int y);
int main()
{ int x, y, a;
  printf("inserisci due numeri");
  scanf("%d %d", &x, &y);
  a = exp(x, y);
  printf("%d", a);
  return 0;
}
int exp(int x, int y)
{ int r = 1, i;
  for(i=0; i<y; i++)
    { r = r*x; }
  return r;
}
```



passaggio parametri per indirizzo:
↳ passo a f(x) non valore della mia var indirizzo.
così f(x) può scrivere su variabile che è a main
se passo indirizzo (vettore) → f(x) può modificare vettore

Indirizzo
scalari → &a
vettori → vet

printf("%d", &a)
no parentesi, indirizzo vet[N]
dem. vettori: &v[0]

vettori possono essere argomenti funzioni

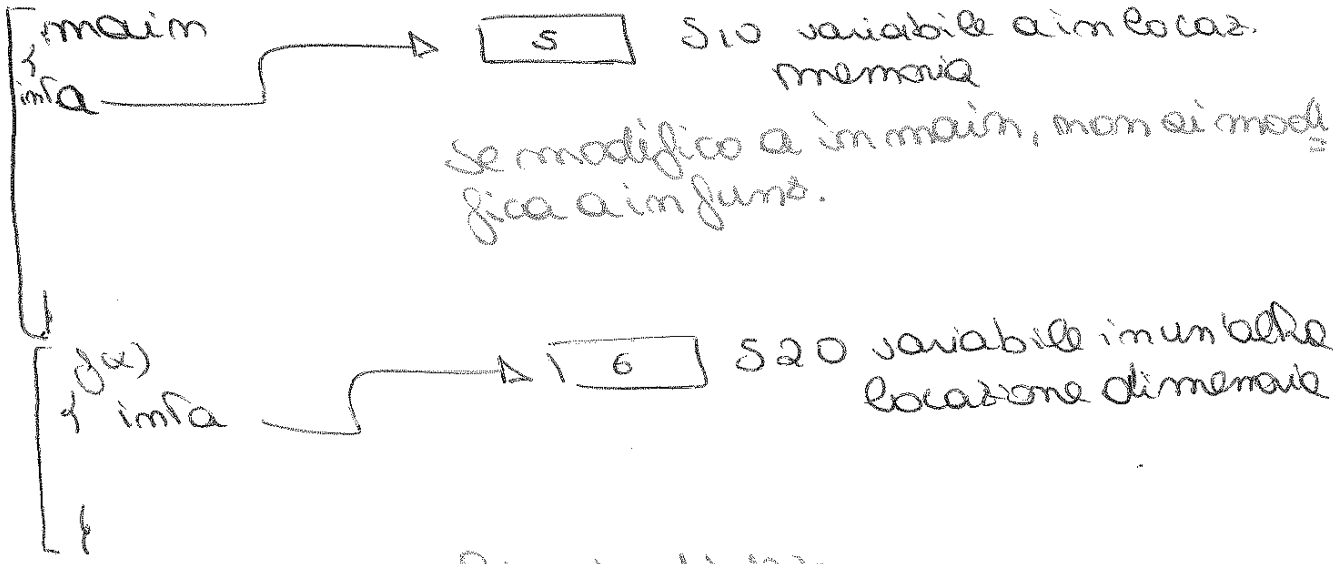
inf f(int v[], int dim);
vettore numero elementi

r = f(vi, 3)
↳ vet. senza parentesi gli do l'esso indirizzo

NB: funz. deve sapere dim. vettore

ES: n° elem pari vet
int f(int v[], 3);

↳ gli passo indirizzo vettore.



passaggio parametri per indirizzo:
↳ passo a f(x) non valore della mia var indirizzo.
così f(x) può scrivere su variabile che è a main
Se passo indirizzo (vettore) → f(x) può modificare vettore

Indirizzo
scalari → &a
vettori → vet

printf("%d", &a)
no parentesi, indirizzo vet[N]
elem. vettori: &v[0]

vettori possono essere argomenti funzioni
int f(int v[], int dim);
vettore numero elementi

r = f(v1, 3)
↳ vet senza parentesi gli do l'esso indirizzo

NB: funz deve sapere dim vettore

ES: n° elem pari vet
int f(int v[], 3);

→ gli passo indirizzo vettore.

passo a f(x) anche v[0] e la sua dim.

S_y = sum(& v[0], 3)

```
void quad (int v[], int dim);
main ....
        -> quad(V, N);

void quad(int v[], int dim)
{ int i;
  for(i=0; i<N; i++)
    { v[i] *= v[i]; }
  return;
}
```

ES: vedo scelgo, fa che ordini funzione con output void

```
void ord(int v[], int dim)
{ int i, j;
  int max, pos, temp
  for(i=0; i<dim-1; i++) {
      max = v[i];
      for(j=i; j<dim; j++) {
          if(v[j] >= max) {
              max = v[j];
              pos = j
          }
      }
  temp = v[i];
  v[i] = v[pos];
  v[pos] = temp;
  }
}
```

posso gestire l'indirizzo direttamente

*pa = *pa + 7 = a = a + 7

↳ con asterisco vuol dire "contenuto della variabile a". *pa rimanda ad a

Modo come

ES int a, b;
int *pa, *pb;

a = 3;
b = 7;
pa = &a;
pb = &b;
pa += 7; / aggiunge 7 ad a */

Valuta n° elementi e n° elem dispari.

void f(int v[], int dim, int *pp, int *pz)
(pari) (zero)
int i;

for(i = 0; i < dim; i++){
if(v[i] % 2 == 0){
*pp += 1;
if(v[i] == 0) → opera con indirizzi
*pz += 1;
}
}
return
}

Caratteri memorizzati grazie codice ASCII sul byte
↳ caratteri
↳ numeri

I/O caratteri
getchar() → legge da tastiera 1 carattere
putchar(<carattere>) → stampa a video

NB: per usare funz. gestione caratteri:
include <ctype.h>

Lezione 26 (Esercizi sui caratteri)

Es: dato vett. di caratteri (costante)
leggi 2 caratteri
1 n° caratt. ha i due
2 verifica se presenti e n° volte
3 verifica se 3 sequenze

char c1[N] = {'a', 'm', 'a', 'e', 'z', 'i', 'e'};
char car1, car2;
int i;
printf("immetti 2 caratteri");
car1 = getchar();
car2 = getchar();

→ caratteri hanno ASCII

scanf("%c", &car1);
scanf("%c", &car2);
→ non funziona

come scanf da tastiera
'i' invio
prende due caratteri:
lui digitato e l'invio
che va nel buffer
→ deve "resettare" il buffer

%c%*c → salto questo

scanf("%c%c%*c", &car1);
scanf("%c%c%*c", &car2);

con getchar:
car1 = getchar();
getchar;
car2 = getchar();

1) [illegible]

numero = car1 - car2;
if (numero < 0)
numero = - numero;

[illegible]

for(i=0; i<N; i++)
if (c1[i] == car1)

Lezione 22 (Stringhe)

Stringa: sequenza di caratteri terminata dal '\0' (NULL), serve a capire dove finisce la stringa

char s[10]; "ciao"

char s[10] = "" → [\0 ... \0]

'a' "a"
97 [97|0]
↳ cod ASCII

I/O

gets → input
puts → output

gets(s1);
puts(s1);

scanf("%s", s1) → non richiede &, è già indirizzo
Termina con invio.

pippo bb miao _ invio
gets legge riga
sscanf parola

NB: ricorda che alla fine c'è \0:
char nome[N+1]

Una stringa, quindi, è un array di caratteri terminato da \0

char s1[10] = "miao" giusto.

char* strcat(char* s1, char* s2) concatena s1 e s2
char* strchr(char* s, int c) ricerca c in s
char* strcmp(char* s1, char* s2) confronto: 0, >0, <0
char* strcpy(char* s1, char* s2) copia s2 in s1
char* strlen(char* s) [illegible]

sscanf(<nomestr>, "%<tipo><carat. guida>%<tipo> ---"
<var1>, <var2> ----)

sscanf(s, "%s %d", s1, &var);

pi = (var - 130) * 0,75 + 130;

↳ posso usare operazioni aritmetiche (float)

Lezione 28 Es. stringhe

sscanf: suddivide stringa in "pezzi" di diverso tipo.
sprintf: parte da + variabili e le mette in una sola stringa

int sprintf(char* <stringa>, char* <formato>, <variabili>);

Es: leggi 2 stringhe.
- se uguali = invertono con s2, concatenano i e s2
- se ≠ le scambio

```
#include <string.h>
{ char s1[30+1], s2[20+1], temp[30+1];
  gets(s1);
  gets(s2);

        }
    // scambio elementi
    for(i=0; i<strlen(s2); i++){
        temp = s2[i];
        s2[i] = s2[strlen(s2) - 1 - i];
        s2[strlen(s2)-1-i] = temp;
    }
```

```
else {
    strcpy(temp, s2);
    strcpy(s2, s1);
```

uso stringa temporanea!!

```
for(i=0; i<strlen(S); i++) {
    k = S[i] - 'a';
    V[k]++;
}
```

* S[i] - 'a' → se = 0 → allora S[i] è a, ma dà posizione dalla lettera nel vettore delle alfabetiche!!

```
for(i=0; i<26; i++) {
    if (V[i] != 0)
        printf("lettera %c trovata %d", 'a'+i, V[i]);
}
```

'a'+i → lettera a i-esima posizioni dalla lettera 'a'

NB: programma funz. con lettere minuscole.

Lezione 29 (Martedì)

↳ array multidimensionali

Es: introduco serie stringhe fine con stringa "fine"

1) qualche max n° cifre

Dobbiamo memorizzarle tutte?

ci serve vettore di stringhe!

ma dato fisso di stringhe che ai posti non interessa posso non memorizzarle, in memoria metto solo n° cifre in V[i]. e la stringa con n° max cifre

```
#include <string.h>
#include <ctype.h>
#define N 20
int main()
{ char S[N+1], Smax[N+1];
```

stringa che leggo — stringa che memorizzo

int i, max, cont;

m[i][i] → diagonale principale
m[i][N-i] → diag. secondaria
NB: matrice quadrata

```
{inf m[R][R]= { {0,2,3}, {1}, {4,5,6} };
 inf i, sum=0;
```

(→ completabile con)

Dichiarazione e inizializzazione matrice:

```
inf <nome>[R][C]= { {vett1}, {vett2}, ... }
```

```
for(i=0; i<R; i++){
    sum += m[i][i];        diag. principale.
}
printf("diag = %d", sum);
sum2 = 0;
for(i=0; i<R; i++){
    sum2+=m[i][R-i];
}
printf("diag2 = %d", sum2);
```

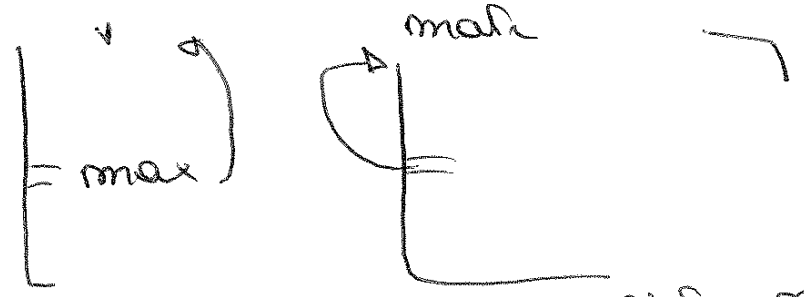
Lezione 30 (Es. su matrici)

Es: introduco mat, vett. che ogni riga mat contiene elem > soglia

```
{inf m[R][C], v[R];
 inf i,j;
 for(i=0; i<R; i++){
    for(j=0; j<C; j++)
        scanf("%d", &m[i][j]);
 }
```

In put matrice da tastiera

```
media = (float) sum/N;  casting
printf("media = %.f", &media);
mx = ael[0];
for(i=0; i<N; i++){
    if( ael[i] >= max){
        mx = ael[i];
        posm = i;
    }
}
printf("Incidenza: %.5 alfmax = %.d", m[posm], &mx);
```



Servono due variabili temporanee: 1 stringa e 1 intero.

NB: devo usare strcpy e non tmp=s

Lezione 31 (Es matrici e Aree di comando)

Es: matr. 3 2x2 con somma elem = 0

```
int m[R][C] = {{1,2,0,0},{0,0},{12},{34}};
int i,j,som,k,r;
for(i=0; i<R-1; i++){
    for(j=0; j<C-1; j++){
        som=0
        for(k=i; k<i+2; k++){
            for(r=j; r<j+2; r++){
                som+=m[k][r];
            }
        }
    }
}
```

cambiano a passo cambiano matrici mxn

F: (disco, removibile)

F:\> dir → fa elenco cartelle

F:\programmi: in cartella "programmi" ci F

obiettivo: dare input a progr. in C dalla linea di comando, do argomenti

main(void)

↳ non ricevo da sist. operativo nessun parametro

Se ricevo parametri

↳ n° param

↳ param.

main(int argc, char* argv[])

ex 23 3,2 => 3, 2 + 1 → nome progr

2 param (pippo)

↳ li mette in vett. stringhe argv

argv[0] → nome progr.

argv[1] → 1° parametro

argv[2] → 2° parametro

C:\progr> quadrato 5

nome progr parametro

argc = 2

NB - argomenti separati da spazi

argv[0] = "pippo.exe\0"

ES: 2 numeri interi

2 lampi quadrato.

```
main(int argc, char* argv[])
if (argc != (2+1)) {
        printf("errore\n");
        return -1;
}
else
    sscanf(argv[1], "%d", n1);
    sscanf(argv[2], "%d", n2);
```

sscanf(s, "%s %s %d %d %d", m[i], ma[i], &v1[i], &vm[i], &ve[i]);
↳ spazio.

Controllo: a = sscanf ....
Restituisce la stringa iniziale

if(a != 5)
return(-1); ↳ codice errore

/* max voti fra primo e ultimo anno */

1) r2 [+ ma [genero 3° vettore: r1+ma [ricerco max

↳ ma posso fonderli meno

mx=0; inizializzo
for(i=0; i<N; i++){
sum = v1[i] + v2[i];
if(sum > mx){ calcolo del valore max
mx = sum;
posmx = i;
}
}

printf("studente: %s voto max: %d ", [nome vettore, posizione] m[posmx], &m

2) media voti 1°/ultimo
↳ serve a perdere di casa.
↳ mi serve un vettore? → No, non mi serve

for(i=0; i<N; i++){
media = (float)(v1[i] + vma[i])/2;
printf("studente: %s media: %s media: %f", m[i], ma[i], &media);
}

vg[i+1] = tt;

{ if

1 2° for

} 3° for

poi da capo come for un printf.

Lezione 33 (Esercizi main. e stringhe)

{fine es lez. 32.}

5) leggi da linea di comando un valore, stampa a stud con vg > soglia

leggere da linea di comando:

main (int argc, char* argv)

posso eseguire programma anche se calcolabr senza codeblocks, basta prendere file.exe

se non lo scrivo, non funz. via linea di comando

stud: m[ ][ ]

vet: vg[ ]

come acquisisco param. da linea di comando?

if (argc != 2)
return -1; } controllo input argomenti

argv[0] = n° progr.

argv[1] = soglia (come stringa, devo trasformarla in intero!!)

sscanf (argv[1], "%d", &soglia)

trasforma stringa in scalare

valore, non vettore

```
if (argc != 2) {
    printf ("ERRORE\n");
    return -1;
}
sscanf (argv[1], "%d", &soglia);
```

uso argc e argv come se fossero variabili

funzione:

```
int somma(int m[][c], int k, int q) {
    int somma, r, w
    somma = m[i-1][j-1] + m[i][j-1] + m[i+1][j-1] +
          + m[i][j] + m[i][j+1] + m[i-1][j]
          + m[i-1][j+1] + m[i+1][j] + m[i+1][j+1];
```

oppure con il for:

```
somma=0;
for(r=k-1; r<(k-1)+3; r++){
    for(w=q-1; w<(q-1)+3; q++){
        somma += m[r][w];
        {
    {
return somma;
{
```

→ richiedo somma minima

```
* if (sum < min){
      min = sum;
      imin = i;
      jmin = j;
  {
```

i,j coordinate elem. centrale.

NB: min = INT_MIN: già inizializzato.

```
printf("il minimo è %d in i=%d e j=%d", min,
       imin, jmin);
return 0;
{
```

p1.età = 13;
strcmp(p1.nome, "marco");
è stringa, con strcmp copio in "scolastica"
nome ∈ p1 "marco"

Typedef <tipo> <nome nuovo tipo>;

typedef struct {
double re;
double im;
} compl

nome comodo al gruppo, nell'altro modo per def. la struttura il nome comodo all'inizio

Es: Due pti in un piano →
leggo da tastiera
1°) centro cerchio
2°) pto circ
perimetro e Area

p1.x p1.y
p2.x p2.y

int main()
{

int raggio,

printf("p1");
scanf("%f %f", &p1.x, &p1.y);
printf("p2");
scanf("%f %f", &p2.x, &p2.y);
raggio = sqrt((p2.y - p1.y)*(p2.y - p1.y) + (p2.x - p1.x)
* (p2.x - p1.x));

.... nel main:

```
if (v==1)
    printf("equilatero");
else
    printf("no equilatero");
return 0;
```

Non devo dichiarare la funzione "printf" perché è già nel main, è una funzione di sistema.

```
#include<stdio.h>
#include<stdlib.h>
struct punto{
    float x;
    float y;
};
infequi(punto P1, punto P2, punto P3);
```

→ dichiaro tipo punto

Lezione 35

File gestione informazione su memoria di massa

Info su disco
↳ settori in 512 byte

File → Record, app. di campi, ecc. da programmi applicativi;

File ha un qualificatore (estensione)
↳ dice programma da associare a informazione

File ASCII → testo puro .txt (Blocco note)
info stile non come app. caratteri → testo formattato (.html .csv .rtf), info in binario interpretate da office Word. Struttura = evoluta, conformato.
→ linguaggio puro (.c)

File binario → proprietario
↳ standard (.pdf, .zip, .jpeg)

<....> codifica formato.

File sequenziali (stream) sequenza di byte, ognuno separato da presenza di codice, termina con EOF
non si può saltare da 1 ob all'altro

per cambiare cartella dove è file:

f = fopen("c:\\prog\\dati\\x", "r");
specifico percorso

"r" solo lett.
"w" solo scritt (scrive da inizio)
"a" agg. in coda a file (scrive da fine)
"r+" "w+" "a+" → su nuovo file
mano a mano che leggo punt. si sposta
int fclose(FILE* <file>)
ES: fclose(ff) fine connessione file fisico - file logico
0 → operaz. corretta
carattere → errore

fclose(ff)
↳ variabile stream

Come si manipola:
- per caratteri
- per stringhe

Lettura a caratteri:
int getc(FILE* <file>);
int fgetc(FILE* <file>);
↳ legge carattere dove è posizionato punt. re, la toglie 1° carattere, poi sposta di +1 il puntatore, ritorna codice ASCII del carat.

int fputc(int c, FILE* <file>);
fputc('m', ff)
↳ scrive m dove c'è punt. (1° carattere) poi sposta punt.

fgetc e fputc → 1 solo carattere
fgets e fputs → stringa

while (fgets(....) != NULL)

oppure:

while (fscanf(....) != EOF)

ES: file di testo, brani progr che stampi freq. varie lettere + nome file da linea comando.

↳ vettore di frequenze

- posso leggere per carattere: fgetc
- leggere per righe: fgets

sottraggo cod ASCII prima lettera, avrò pos. in vett. delle frequenze da incrementare.

a) per caratteri:

```
const int LETT = 26;
int freq[LETT];

FILE *f;
int ch, i;
if (argc != 2)        /* da linea di comando */
  {
    printf ("n° argom. errato \n");
    exit(1);
  }

for(i=0; i < LETT; i++)
      freq[i] = 0;
f = fopen(argv[1], "r");
ch = fgetc (f);
while (ch != EOF)
  {
    if (isalpha(ch))
      { i = toupper(ch) - 'A';
```

argc: n° argomenti, nome progr e nome file = 2

argv[1] → nome file, argv[0] = nome progr

"r" → modalità lettura

→ lettura carattere per carattere.

```
FILE *f;
f=fopen("pippo.txt","w");   → voglio scrivere
                              ho aperto il file in write, inizio a scrivere dal 1° carattere
if (f==NULL){
printf("errore!");
return-1;
printf("stringa");
scanf("%s",s);
printf("** age **");
scanf("%d",&age);
fprintf(f,"%s %d",s,age);
```

fscanf
fgets
leggo da tastiera
scrivo su file
fputs
fprintf

→ scanf in formato <stringa> <age> separati spazio num

```
fclose(f);
```

Calcolo min: uso solito loop.

```
f=fopen("nomefile","r");  devo aprirlo in lettura;
if(f==NULL)
{printf("errore");
return-1;
}
while(fscanf(f,"%s %d",s,&age)!=EOF   → nomefile, → metto in age
{if(age<=min)                          → metto in s
{min=age;
strcpy(&min,s);   → copio la stringa in cache
}
}
```

NB: min=INT_MAX; altrimenti inizializzato

```
printf("nome=%s alt=%d",&min,&min);
```

```
if(f==NULL){                    → controllo apertura file
  printf("errore");
  exit(1);                      → programma termina con errore, exit(0), non ha errori
}
/*inizializzo matrice rettangoli*/
int i,j;                        → sono N+1 righe/colonne
for(i=0;i<=N;i++){              con 2 cicli for inizializzo 20 righe e colonne.
  for(j=0;j<=N;j++){
    matrice[i][j]='0';
  }
}
```

fgets(s, 70, f) → f file, 70 max di miga

↳ dove metto stringa

S+1 caratteri → '\0', termina la stringa

9 | 10 | 99 | 1

2 x 4 + 3 spazi

11 caratteri + 1 '\0'

```
/* occupare la matrice */
for (i = y1 ; i <= y2 ; i++) {
    for (j = x1 ; j <= x2 ; j++) {
        matrix[i][j] = '1';
```

cambio valore delle celle di matrice da '0' a '1' perdire se è occupata.

↳ sovrapposizione
int flagsovrapposto = 0;
dichiaro all'inizio.

↳ se diventa = 1, deve terminare il ciclo, lo metto in condiz. esterna

```
for (j = x1 ; j <= x2 && flagsovrapposto == 0 ; j++)
... lo aggiungo anche prima!
for (i = x1 ; i <= y2 && flagsovrapposto == 0 ; i++)
while (gets ..... != NULL && flag == 0)
```

```
/* stampa messa finale */
if (flagsovrapposto == 0)
    printf ("non ci sono sovrapposizioni");
else
    printf ("c'è almeno una sovrapposizione");
fclose (f);
```

NB: ricordarsi di chiudere il file, se si dimentica quando si scrive viene persa ogni modifica

3) algebra booleana:

$xy + xyz = xyz$

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

le due espressioni non sono equivalenti

Lezione 38

Esercizi Teoria

1) F3 1A

A B C D E F
10 11 12 13 14 15

F3

$F: 2^3 + 2^2 + 2 + 1$

1111

a) 11110011

$A: 2^3 + 2$

b) 00011010 c.d. $-2^7 + 2^6 + 2^5 + 2^4 + 1 + 2 = -128 + 1$

complem a 2 → no overflow segno ≠!

$$\begin{array}{r} 1111\,0011 \\ 2 \quad 0001\,1010 \\ \hline 0000\,1101 \end{array}$$

cod. normale → overflow

es22: 243 (a)
26 (b) (anche c.2)

5) $x + y'x' = xy' + x'y$

x	y	y'	x'	$x'y'$	$x + x'y'$	xy'	$x'y$	$xy' + x'y$
1	0	1	0	0	1	1	0	1
0	0	1	1	1	1	0	0	0
0	1	0	1	0	0	0	1	1
1	1	0	0	0	1	0	0	0

Sono diverse.

Ex 1 mappa terreno verde urbana fluviale

ogni area → h in m

riga: <area> <h> < ... > < > < > < > 4 aree

max 5 righe

1) [illegible]

2) [illegible]

3) [illegible]

4) [illegible]

[illegible]

area → tipo — che struttura dati uso?

↳ alloco

2 matrici associate

int mm[R][4]

char m[R][4]

```
#include <stdio.h>
#include <stdlib.h>
#define R 5
#define C 4

int main(int argv, char* arg[c])
{ FILE *fp;
  char [R][C];
  int [R][C];
```

TEORIA

1) Illustra la funzione dell'address bus, e spiega quali sono le influenze della sua dim.

Max memoria indirizzabile =

$2^{Abus} \times Dbus$ bit.

l'Abus è un bus del calcolatore usato per specificare l'indirizzo di un dato, la sua grandezza determina la q. di memoria indirizzabile.

2) Illustra la funzione della floating point unit e la sua influenza sulla potenza di calcolo

La FPU, unità di calcolo in virgola mobile è una tipologia di processore specializzata in calcoli fatti con la virgola mobile. Questo processore è in grado, che alle semplici addizioni, di svolgere calcoli + complessi come radici o coseni.

3) Spiega cos'è l'area di SWAP.

L'area di SWAP è una zona di memoria sul disco fisso, con il suo utilizzo si realizza la cosiddetta virtualizzazione della memoria. Si fa in gatti "credere" ai programmi che la memoria disponibile sia molto più grande

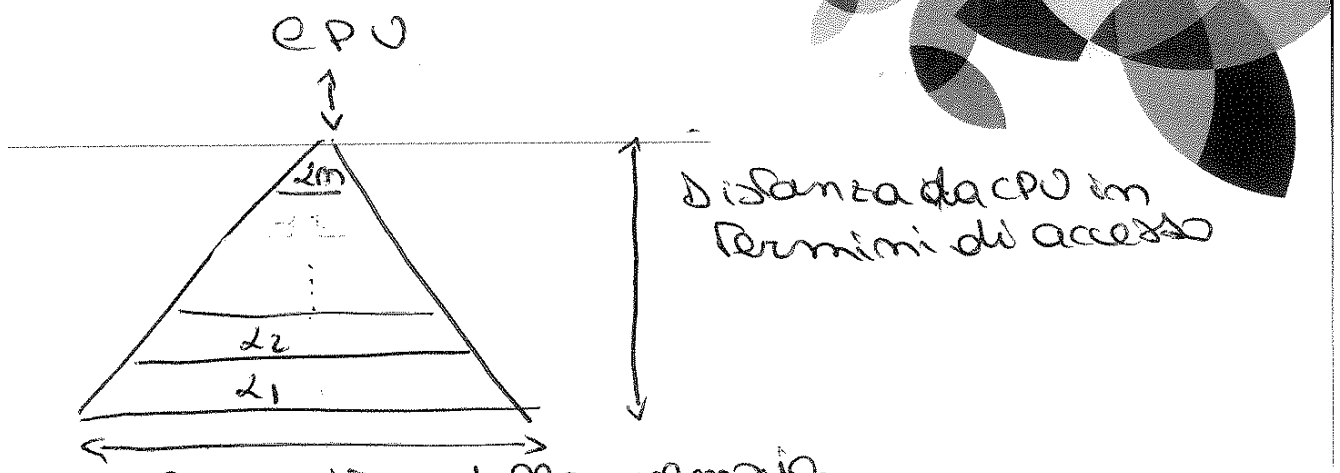
4) Disegna lo schema di una generica unità operativa

Un esempio di unità operativa è la CPU.

[Gest. dec. istr.] [Gestione Bus]

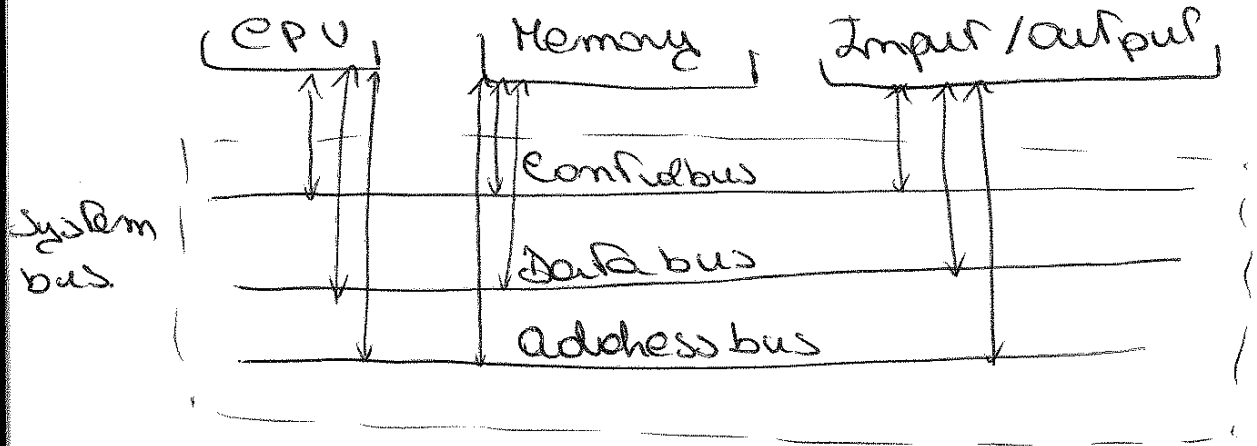
[Gest. indir.] [U. controllo] [Registri] [ALU]

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 95 di 116



In questo schema Lm è la + veloce, contiene solo i dati e le istruzioni "+ vicine" a quella in esecuzione

8) Illustrare quali sono i tipi di Bus dell'architettura di Von Neumann



9) Descrivere l'importanza di registri e flag in un elaboratore.

I registri sono locazioni interne alla CPU alle quali è possibile accedere molto velocemente. Ad esempio il registro IP contiene l'indirizzo della prossima istruzione da eseguire.
I flag segnalano particolari stati della CPU, i + importanti sono:

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 97 di 116