RIASSUNTI ESAME ORALE

▼ ALGORITMI ED ESECUTORI

▼ Che cosa si intende per informatica?

- **Scienza della rappresentazione ed elaborazione dell'informazione.** L'informazione, dunque, risulta essere il concetto principale dell'informatica, mentre per essere elaborata si considera lo svolgimento atto in maniera automatica tramite un **calcolatore (o computer).**
- **Scienza dell'astrazione,** ovvero serve a creare il giusto modello per un problema e ad individuare le tecniche appropriate attue a risolverlo in modo automatico. L'obiettivo risulta quindi essere quello di voler ricreare una situazione reale, abbastanza complessa, e renderla semplice e comprensibile, entro la quale si possa risolvere il problema.
- **Definizione finale:** obiettivo dell'Informatica è creare delle astrazioni di problemi del mondo reale che possano essere rappresentate ed elaborate all'interno di un sistema di calcolo al fine di eseguire dei procedimenti risolutivi in modo automatico.

▼ Che cos'è un algoritmo?

- **Definizione Algoritmo:** è un procedimento sistematico, costituito da un insieme finito di operazioni, precise ed eseguibili, da applicare ai dati in ingresso purché possano fornire dei dati in uscita.
- **In un algoritmo è possibile stabilire** sia la fine del procedimento, sia il numero di passi compiuti, più precisamente un numero finito di passi, e sia quale operazione si debba compiere in quel preciso momento. Una volta definito, l'algoritmo deve essere sottoposto ad un **esecutore** che deve essere in grado di:

1. memorizzare informazioni che permettano di accedere ad esse ed anche di modificarle.
2. eseguirli;
3. interpretare la sequenza di comandi.

▼ Algoritmo e programma

- La descrizione di un algoritmo è indipendente dall'esercutore che dovrà eseguirlo. La sua rappresentazione, comprensibile ed eseguibile in automatico dall'esecutore, costituisce un **programma**.
  L'elaborazione delle azioni viene definita tramite comandi elementari chiamati **istruzioni** espresse tramite il **linguaggio di programmazione.**
- Un programma, quindi, è la formulazione testuale, in un certo linguaggio di programmazione, di un algoritmo che risolve un problema. Esistono vari tipi di linguaggi di programmazione (Python, Java, C, Pascal...) per questo un algoritmo può essere implementato in linguaggi diversi, di fatti ognuno dei programmi ottenuti è equivalente agli altri essendo un'implementazione dell'algoritmo originale.
- L'uso di un linguaggio di programmazione permette di:
  1. realizzare un programma che implementa un algoritmo in maniera precisa ed in un linguaggio ad "alto livello", cioè molto vicino a quello umano;
  2. realizzare un programma che non dipende dal calcolatore su cui viene eseguito.

▼ RAPPRESENTAZIONE DEGLI ALGORITMI E STRUTTURE DI CONTROLLO

▼ Concetto di informazione

- Definizione: per informazione si intende tutto ciò che può consentire di ridurre il grado di incertezza in merito ad una particolare situazione. Quindi per scambiare un'informazione si pressupone l'esistenza di due entità: un mittente, il quale fornisce l'informazione, e un ricevente, il quale la riceve. In un algoritmo, l'informazione assume quindi tre caratteristiche fondamentali:

- Valore: indica l'elemento assunto dall'informazione.
- Tipo: indica l'insieme degli elementi nei quali fa parte il valore.
- Attributo: indica il contesto nel quale si trova l'informazione.
- Si ottiene, dunque, un'informazione quando un attributo assume un valore di un determinato tipo.

▼ **Concetto di variabile**

- Una variabile è un ente la quale appartiene ad un determinato tipo ed essa è identificata da un nome che rispecchia, il più delle volte, il ruolo che essa assume all'interno dell'algoritmo. Il valore di una variabile può essere modificato.

▼ **Concetto di costante**

- La costante e' un oggetto, appartentente ad un tipo, il cui valore rimane invariato per tutta l'esecuzione dell'algoritmo ed essa può essere identificata da un nome.

▼ **Concetto di espressione**

- Un'espressione è un insieme di operatori ed operandi (variabili, costanti) ed il tipo di un'espressione dipende dal tipo degli operandi coinvolti in essa.

▼ **Concetti di Strutture di Controllo**

- Sono costrutti elementari per descrivere in maniera completa la parte esecutiva di un algoritmo.

▼ Esse si suddividono in:

- sequenza di operazioni (assegnazioni di valori a variabili);

▼ selezione (di azioni diverse in base alla valutazione di una condizione):

1. Selezione semplice
2. Selezione a due vie

▼ esecuzione ciclica (non si definisce esplicitamente il numero di ripetizioni, ma la fine del ciclo dipende da una condizione che, finchè

risulta vera, causa un'altra ripetizione del ciclo:

1. Ciclo a condizione iniziale (while): viene eseguito un minimo di 0 volte.
2. Ciclo a condizione finale(do, while): viene eseguito un minimo di 1 volte.

▼ Operazioni di input e output:

- con le operazioni di input, il valore di una variabile viene modificato grazie ad un'operazione di lettura dall'unità di ingresso (tastiera);
- con le operazioni di output, viene valutato il valore di un'espressione e viene presentato sull'unità di uscita (schermo).

▼ ALGEBRA DI BOOLE

▼ Definizione

- è un'algebra in cui le variabili e le funzioni possono assumere solo due valori: true e false (vero, falso).

▼ Componenti Booleani

- operatori booleani: legano insieme le espressioni logiche;
- gli operandi booleani: sono valori logici che possono assumere solo true e false come valori (vero, falso).

▼ Operatori

- and (congiunzione,operatore binario)
- or (disgiunzione, operatore binario)
- not (negazione, operatore unario)

▼ Combinazioni (A, B, variabili booleane)

- A and B==TRUE se e solo se sono entrambi TRUE, altrimenti restituisce FALSE (moltiplicazione)
- A or B==FALSE se e solo se sono entrambi FALSE, altrimenti restituisce TRUE (addizione)

- not A==TRUE se A è FALSE (opposto)

▼ Precedenza degli operatori

- E' possibile costruire un'espressione in cui ci siano più operatori. In questo caso e in assenza di parentesi, la precendeza è: NOT, AND, OR.
- Contraddizione (A and not A)
- Tautologia (A or not A)

▼ RAPPRESENTAZIONE DEI DATI

▼ Memoria e quantità di informazione

- La più piccola unità di informazione memorizzabile è il bit che assume 0 o 1 come valori. Per memorizzare un bit si usa un elemento bistabile, cioè un dispositivo il quale può assumere due stati differenti che vengono fatti corrispondere a 0 o ad 1 (cella di memoria);
- Sono possibili due tipi di operazioni sulla cella di memoria:
  1. Operazione di lettura: si accede alla cella per consultarne il valore e copiarlo su un'altra cella di memoria.
  2. Operazione di scrittura: la cella di memoria viene caricata con un valore (0 o 1) che rimane memorizzato fin quando non viene sovrascritto;

▼ Registro di memoria

- Un insieme di N celle può assumere uno fra 2^N stati possibili. Tale insieme è detto registro di memoria. Un insieme di 8 bit è detto byte.
- Il registro permette di memorizzare una tipologia di informazione il cui valore fa parte di un insieme finito e discreto.
- Introduciamo il concetto di codifica essendo che il registro di memoria consente di memorizzare diverse tipologie di informazioni: troviamo quindi una rappresentazione del dato gestibile con le possibilità offerte dall'elaboratore.

▼ La codifica

- Un'informazione può essere rappresentata in diversi modi, ma quest'ultima deve essere comprensibile al destinatario che la riceve.
- Un sistema di codifica usa un insieme di simboli, come l'alfabeto, o combinazione di essi.

▼ Rappresentazione dei dati

▼ Rappresentazione dei numeri

- E' un sistema posizionale e pesato. Esempio: (3707 = 3 × 103 + 7 × 102 + 0 × 101 + 7 × 100);
- Base di numerazione 10: cifre=(0,1,2,..9), pesi=(potenze di 10);
- Base di numerazione 8: cifre=(0,1,2...7), pesi=(potenze di 8);
- Base di numerazione 2: cifre=(0,1), pesi=(potenze di 2);

▼ Rappresentazione dei caratteri

- Si utilizza la tabella ASCII

▼ LINGUAGGI DI PROGRAMMAZIONE E LINGUAGGIO C

▼ Linguaggi di programmazione

- Anche le istruzioni, così come le informazioni, si possono codificare come sequenze di bit. Nello specificare un'istruzione bisogna precisare l'operazione da compiere e i dati coinvolti in essa.
- Essendo l'insieme delle diverse operazioni un insieme finito è possibile codificarlo con un certo numero di bit (codice operativo) e quindi un esecutore permette l'esecuzione di un programma, cioè di una sequenza di istruzioni che realizzano un particolare algoritmo e che sono descritte nel linguaggio intepretabile dal calcolatore.

▼ Vantaggi

- si può realizzare un programma che implementa l'algoritmo tramite un linguaggio ad alto livello;
- si può definire un programma che non dipende dal calcolatore sul quale è stato realizzato;

    - si possono trascurare i dettagli relativi alla rappresentazione dei dati nei registri.
- I linguaggi di programmazione ad alto livello sono linguaggi in cui la sintassi e la semantica sono definite tramite regole rigide e precise, così da eliminare le ridondanze tipiche del linguaggio comune e così da tradurre (tramire i compilatori) un programma scritto in un linguaggio ad alto livello ad uno in linguaggio macchina.

▼ Linguaggio C

▼ Tipi

- Tipo Int: costituito da un sottoinsieme limitato di numeri interi; ha dimensione 4 bytes e sono ammesse tutte le operazioni.
- Tipo Float: costituito da un sottoinsieme limitato e discreto di numeri reali; ha dimensione 4 bytes e sono ammesse la maggior parte delle operazioni.
- Tipo Double: costituito da un sottoinsieme limitato e discreto di numeri reali, ma con più precisione rispetto al float; ha dimensione 8 bytes e sono ammesse la maggior parte delle operazioni.
- Tipo Char: costituito da un insieme di caratteri, alcuni stampabili ed altri non; per la loro rappresentazione viene usata la tabella ASCII; ha dimensione 1 byte e sono ammesse le operazioni tipiche degli interi.
- Tipo Bool: costituito da soli due valori (false e true) rappresentati da 0 e 1; ha dimensione 1 byte e sono ammesse le operazioni booleane (=; | |; & &; !).

▼ Variabili

- Per poter utilizzare una variabile essa deve essere definita. Con la definizione viene creato uno spazio in memoria adeguato per ospitare una variabile di quel tipo; non può essere spostata su un altro spazio di memoria e possiede un valore casuale.
- Essa mantiene il tipo assegnato fino alla fine del programma.

▼ Costanti

▼ Costanti letterali

- Il valore della costante è rappresentato direttamente.
- Costanti di tipo int: sono definite come sequenze di cifre decimali.
- Costanti di tipo float: sono definite come sequenze di cifre decimali, strutturate in virgola fissa o mobile (0.1; -3.7; 1.0E4).
- Costanti di tipo char: sono definite come caratteri racchiusi tra singoli apici.
- Costanti di tipo stringa: sono definite come sequenze di caratteri racchiusi tra doppi apici.
- Costanti di tipo bool: sono solo due (true, false).

▼ Costanti definite

- Si utilizza la direttiva **#define** che associa testualmente un valore ad un identificatore. All'atto della compilazione il preprocessore sostutuisce l'identificatore con il valore corrispondente.

▼ Costanti dichiarate

- Il valore viene associato ad un identificatore e ne viene specificato il tipo
- L'identificatore, in questo caso è, a tutti gli effetti, una variabile non modificabile.

▼ Operatori

- Un operatore specifica un'operazione da eseguire su uno o due operandi definendo un'espressione

▼ Espressioni

- E' una combinazione di operandi e operatori. La sua valutazione porta al calcolo di un valore, appartentente ad un certo tipo. Nel caso vi siano più tipi l'espressione assume il tipo "più ampio" tra quelli presenti.

- Casting implicito: non necessitano di alcun operatore ma sono eseguite in automatico.
- Casting esplicito: è' possibile modificare il valore del tipo di un'espressione indicando esplicitamente il tipo da impiegare.

▼ LINGUAGGIO C - I/O E COSTRUTTI

▼ Istruzioni di calcolo e assegnazione

- Abbiamo già visto operazioni di calcolo e assegnazione, come, ad esempio, per le espressioni il cui valore veniva assegnato ad una variabile.

▼ Operazioni di Input/Output

- Esse sono dichiarate nella libreria **stdio.h**

▼ Input

- Con le operazioni di input il valore di una variabile viene modificato con il valore ottenuto grazie ad un'operazione dall'unità di ingresso.
- Per le operazioni di input si utilizza **scanf**(stringa di formato, elenco variabili).
- La funzione scanf legge caratteri dallo stream di input (tastiera), li converte in dati di un certo tipo secondo la stringa di formato e i dati vengono memorizzati nelle variabili dell'elenco variabili.
- L'elenco delle variabili conterrà gli identificatori delle variabili assegnate a scanf; più precisamente verrà fornito l'indirizzo di memoria di ogni variabile nel quale essa è memorizzata (&).

▼ Output

- Con le operazioni di output il valore di un'espressione viene presentato sull'unità di uscita.
- Per le operazioni di output si utilizza **printf**(stringa di formato, altri argomenti), dove la stringa di formato può contenere specificatori di formato. Questi ultimi devono essere in egual numero agli argomenti.

- Specificatori di formato
  - %d interi;
  - %s stringhe;
  - %c caratteri;
  - %f numeri reali.

Strutture ci controllo

Sequenza

- La sequenza è costituita da un insieme di istruzioni successive le quali vengono eseguite nell'ordine in cui compaiono.

Selezione

- I costrutti di selezione permettono di scegliere di eseguire una tra due istruzioni in base alla valutazione di una condizione (if; if..else; if..else if...else);

Ciclo

- I costrutti di ciclo servono a ripetere l'esecuzione di un'istruzione.

Costrutti di ciclo a condizione

- **Ciclo while** è un ciclo a condizione iniziale. Non si definisce il numero di ripetizioni, ma si valuta all'inizio del ciclo una condizione che fin quando risulta vera il ciclo si itera (numero minimo di esecuzioni=0);
- **Ciclo do..while** è un ciclo a condizione finale. Non si definisce il numero di ripetizioni, ma si valuta alla fine del ciclo una condizione che fin quando risulta vera il ciclo si itera (numeri minimo di esecuzioni=1);

Costrutti di ciclo a conteggio

- **Ciclo for** è un ciclo a conteggio. Si definisce esplicitamente il numero di ripetizioni e il conteggio viene gestito da una variabile, che assume un valore iniziale, che viene incrementata fin quando non arriva ad un valore finale.

TIPI STRUTTURATI - ARRAY

Definzione

- In alcuni casi l'informazione che bisogna elaborare consiste in un aggregazione di valore piuttosto che di un valore solo. Per indicare l'insieme di valori che ci interessano si utilizza l'Array.
- Un array è quindi un insieme di variabili tutte dello stesso tipo identificato da un nome unico e i suoi elementi sono disposti in memoria in posizione consecutive.
- Per definire un array è necessario specificare: il nome e il tipo della variabile array; il numero degli elementi presenti (cardinalità).

Acceso agli elementi dell'array

- Per accedere ai vari elementi dell'array è necessario specificare il nome della variabile array e la posizione dell'elemento di interesse che si definisce indice.
- L'indice parte da 0, quindi v[0], ad esempio, sarà il primo elemento del vettore. Se si definisce un array di N elementi, l'indice andrà da 0 a N-1. Il controllo dell'indice è a discrezione dell'utente, quindi se durante l'esecuzione dovesse essere presente un elemento in posizione v[N] sarà effettuato un accesso a zone della memoria che non appartengono all'array.

Dimensione, lettura e stampa

- La dimensione dell'array deve essere una costante.
- Molte volte si usa una dimensione che si ritiene adeguata ad ospitare il numero massimo di elementi e si affianca all'array una variabile intera che contiene il numero effettivo di elementi memorizzati (riempimento).
- Per inizializzare da input una variabile array è necessario utilizzare un ciclo for (o do-while), poichè risulta essere l'unico in grado di assegnare da input un valore ad ogni posizione dell'array.

Array bidimensionali

- Esistono anche array bidimensionali i quali, a diffenza degli array monodimensionali visti in precedenza, richiedono un doppio indice (stile matrice).
- Per definire un array bidimensionale è necessario specificare: il tipo e il nome della variabile array; il numero degli elementi presenti nelle due dimensioni.
- Per accedere agli elementi dell'array bidimensionale è necessario specificare il nome della variabile array e gli indici di riga e colonna che individuano quell'elemento.
- Anche per gli array bidimensionali si usano i cicli for e do-while per assegnare un valore ad ogni posizione dell'array.

▼ TIPI STRUTTURATI - STRINGHE

▼ Caratteri

▼ Operazioni con i caratteri

- Ad una varabile di tipo char si può assegnare un qualsiasi carattere (char ch). Le costanti char, invece, sono definite tra singoli apici.
- E' possibile fare operazioni con i caratteri essendo che il linguaggio C li tratta come interi (tabella ASCII). Quando esso viene usato in un'espressione viene valutato il corrispettivo intero del carattere.

▼ Funzioni per i caratteri

- **toupper** restituisce la versione maiuscola dell'argomento (ch=toupper(ch)).
- **tolower** restituisce la versione minuscola.
- Per utilizzare queste due funzioni è necessario includere la libreria ctype.h.

▼ Input e Output

- Con le funzioni scanf e printf invece va utilizzato lo specificatore %c. Inoltre scanf non ignora lo spazio bianco essendo che nella

tabella ASCII corrisponde al valore 32.

- Per l'input e l'output si possono utilizzare anche le funzioni **getchar e putchar** al posto di scanf e printf. Putchar scrive un carattere, getchar legge un nuovo carattere dall'input ad ogni chiamata e questo valore viene assegnato alla variabile ch (ch=getchar()).
- Putchar e getchar sono utilizzati per rendere i programmi più chiari e semplici. Il secondo in particolare è utilizzato in cicli che ignorano caratteri fino al raggiungimento di un particolare carattere.
- Bisogna fare attenzione nell'uso di scanf e getchar. Infatti scanf lascia nel buffer di input i caratteri che ha controllato ma che non ha letto, come il newline. Quindi utilizzando getchar dopo scanf vorrà dire che getchar avrà come valore proprio newline.

▼ Stringhe

▼ Definizione

- Le stringhe sono sequenze di caratteri. Posso essere costanti (letterali) oppure variabili (cioè variabili il cui valore è una stringa). Il terminatore della stringa è '\0'

▼ Stringhe letterali

- Una stringa letterale è una sequenza di caratteri racchiusi fra doppi apici. Sono spesso usate nelle stringhe di formato per printf e scanf.
- Una stringa letterale di lunghezza N caratteri richiede N+1 byte di memoria, poichè all'ultimo carattere va inserito il terminatore \0.
- Una stringa letterale di un carattere è diversa da una costante carattere. Di fatti "a" è rappresentata da un array di 2 char.

▼ Stringhe variabili

- Una variabile stringa è realizzata come un array di char e termina con \0.

▼ Inizializzazione di una stringa

- Una stringa variabile può essere inizializzata al momento della definizione (char date[8]="June 14")
- Se la stringa ha meno caratteri della dimensione dell'array gli elementi in più saranno inizializzati con \0.
- Quindi: lungezza della stringa=numero di caratteri utili (prima di \0); dimensione dell'array=numero di caratteri con cui è definito l'array.
- Se la stringa ha più caratteri della dimensione dell'array quest'ultimo non è utilizzabile come stringa in quanto non è presente un terminatore. Risulta più sicuro omettere la dimensione dell'array stesso.

▼ Input e Output di stringhe

- Per stampare una stringa è possibile usare printf con lo specificatore %s.
- Con scanf ci sono alcuni problemi legati ai blank. Di fatti scanf legge tutti i caratteri diversi dal blank e al primo blank trovato si ferma e aggiunge \0. Per memorizzare anche gli spazi ed eventuali caratteri successivi si può utilizzare un ciclo while con getchar.

▼ Funzioni per le stringhe

- Per utilizzare le funzioni di manipolazione ed elaborazione delle stringhe è necessario introdurre la libreria **string.h**
- strcpy(dest, sorg): utile per copiare una stringa(sorg) in un'altra(dest);
- strlen(): restituisce la lunghezza della stringa;
- strcat(dest, sorg): aggiunge una copia della stringa sorg alla fine di dest;
- strcmp(s1, s2): confronta due stringhe e restituisce:
  - 0 se le due stringhe sono uguali;
  - <0 se s2 precede s1 in ordine alfabetico;

- >0 se s1 precede s2 in ordine alfaberico.

▼ SOTTOPROGRAMMI

▼ Definizione

- Il concetto di sottoprogamma va introdotto poichè nella scrittura di un programma c'è bisogno, molte volte, di replicare un'operazione la quale è gia presente nel codice e quindi, se riscritta ogni qual volta bisogna usarla, può portare ad una minor leggibilità del codice o ad una maggior probabilità di commettere errori.
- L'ideale quindi sarebbe quello di creare un meccanismo il quale con un'istruzione realizzi tale operazione: ciò è realizzato proprio dai **sottoprogrammi**.
- Un sottoprogramma risulta essere una particolare unità di codice la quale non viene eseguita automaticamente, ma soltanto su richiesta del programma principale o di un altro sottoprogramma.
- Per questo motivo il sottoprogramma utilizza variabili proprie, alcune delle quali vengono utilizzate per scambiare dati con il programma chiamante.
- Nel definire un sottoprogramma risulta quindi necessario precisare operazioni e flusso di dati, ovvero quale operazione realizza il sottoprogramma e quali sono i dati in ingresso ed uscita del sottoprogramma.

▼ Struttura di una funzione

- Una funzione è un particolare sottoprogramma che riceve dei dati in ingresso e produce un valore in uscita. Questo valore è calcolato tramite le istruzioni che si trovano nella funzione e che operano sui dati in ingresso.
- Nella funzione sono riconoscibili due parti:
  - **L'intestazione**: riporta le informazioni principali relative alla funzione (nome, tipo restituito, parametri in ingresso).
  - **Blocco**: parte dichiarativa (variabili locali), parte esecutiva (istruzioni).

- La parte esecutiva contiene quindi l'insieme delle istruzioni presenti nella funzione che agiscono sui parametri in ingresso. Al termine c'è un'istruzione di return il cui scopo è sia di terminare l'esecuzione della funzione e sia quella di restituire un valore, tra parentesi, come valore finale della funzione.
- L'esecuzione di una funzione è dovuta da una particolare istruzione del programma che la attiva, detto chiamante, che provoca quindi la sospensione delle sue istruzioni e che riprenderanno dopo che sarà terminata l'esecuzione della funzione.
- I parametri forniti dal programma alla funzione vengono chiamati **parametri effettivi**, ovvero quelli su cui la funzione effettivamente opera. All'interno della funzione questi prendono il nome di **parametri formali**.
- Anche il main è una funzione a tutti gli effetti. Essa viene chiamata dal Sistema Operativo all'atto dell'esecuzione del programma ed il flusso di dati avviene proprio con il Sistema Operativo.

▼ Procedure

- A volte le operazioni da implementare non necessitano la produzione di un valore finale da restituire, ma soltanto l'esecuzione di un'azione come la stampa di valori o la modifica di variabili. Questo tipo di sottoprogrammi prende il nome di **procedure**.
- Nel linguaggio C una procedura viene definita come una funzione di tipo void, ovvero una funzione che non restituisce valori. In questi casi il return finale serve soltanto a terminare l'esecuzione della funzione.

▼ Passaggio per valore

- La tecnica di corrispondenza tra parametri formali ed effettivi viene detta passaggio per valore. Di fatti il valore del parametro effettivo viene copiato nel parametro formale.
- Quest'ultimo costiuisce quindi una copia locale del parametro effettivo, ma ogni modifica fatta su di esso non si riflette sul parametro effettivo di partenza.

▼ Puntatori

- La definizione di una variabile implica l'allocazione di registri di memoria. Alla porzione di memoria si accede tramite l'identificatore della variabile, ma è il complilatore che crea e gestisce la corrispondenza tra la locazione di memoria e l'identificatore della variabile stesso.
- Il linguaggio C dà la possibilità di accedere all'indirizzo di una variabile tramite l'operatore &.
- Inoltre è possibile definire delle variabili di tipo **puntatore** alle quali si possono assegnare gli indirizzi di variabili di un particolare tipo. La definizione di questo tipo di variabili richiede il tipo puntato più un asterisco.
- Tramite i puntatori è possibile accedere alla variabile puntata, sia in lettura che in scrittura.

▼ Passaggio per riferimento

- Nel passaggio per riferimento al parametro formale viene assegnato l'indirizzo del parametro effettivo.
- Così facendo al sottoprogramma è possibile accedere al registro che ospita il parametro effettivo e apportare modifche che saranno visibili anche al programma chiamante.
- Col passaggio per riferimento è quindi possibile creare una funzione che restituisce più di un valore.

▼ Array come parametri

- Gli array sono passati unicamente per riferimento. Come parametro effettivo va fornito solo il nome dell'array, senza specificare altro.

▼ Matrici come parametri

- Nella definizione come parametro formale va definita solo la seconda cardinalità, della prima il compilatore non ne tiene conto. Come parametro effettivo va fornito solo il nome dell'array.

▼ Concetto di visibilità

- Un programma in C può assumere una struttura complessa con l'uso di sottoprogrammi. Questo rende necessario definire delle regole le quali precisano in quali parti del programma è possibile usare un certo identificatore (visibilità).
- L'insieme delle variabili definite in una funzione (parametri formali, altre variabili) prende il nome di ambiente della funzione.
- L'ambiente del chiamante e quello della funzione sono due ambienti distinti, per cui le variabili del chiamante non sono visibili dalla funzione e viceversa.
- Le variabili sono dunque locali, cioè sono utilizzabili solo all'interno della funzione nella quale sono definite.
- La definizione di una variabile definita in un blocco annulla la visibiltà di eventuali variabili con lo stesso nome, ma presenti in blocchi esterni.
- E' inoltre possibile definire variabili globali le quali saranno visibili in tutto il file, tra cui anche le funzioni, dal punto in cui sono definite in poi (**si sconsiglia l'uso**).
- Tramite le variabili globali funzioni differenti possono realizzare uno scambio di dati che non è chiaramente visibile, come invece accade con il passaggio di parametri.

▼ FILE

▼ Definizione

- Le variabili utilizzate fin'ora erano volatili, cioè la loro vita terminava con la fine dell'esecuzione del programma. Con i file è possibile memorizzare dati in forma persistente e che ne permette l'uso in tempi successivi.
- Il file quindi risulta essere un archivio di dati sul quale sono definite le operazioni di scrittura e lettura. Esso viene memorizzato nella memoria secondaria del sistema di elaborazione che ne consente una memorizzazione persistente.

- I file di testo sono file di caratteri, organizzati in linee. Ogni linea termina con un newline (\n), mentre il file termina con la "marca" **End Of File (EOF)**.
- Per accedere al File da un programma in C è necessario predisporre una variabile che lo rappresenti (puntatore al File).

▼ Apertura e chiusura File

- Prima di accedere ad un file è necessario aprirlo (fp=fopen). Una volta aperto su di esso sono possibili le operazioni di scrittura e lettura.
- Alla fine di una sessione di accesso è necessario chiudere il file per memorizzare permanentemente il suo contenuto in memoria di massa.
- Un file può essere aperto in più modi:
  - "r", read, in lettura (il file deve essere esistente, altrimenti restituisce NULL).
  - "w", write, in scrittura (se il file esiste i suoi contenuti si perdono, altrimenti viene creato).
  - "a", append, scrittura aggiunta alla fine.

▼ Input e Output di un File di testo

- Possiamo realizzare sui file le operazioni di input e output. Tutte le funzioni già definite acquisiscono una "f" davanti al nome (fscanf, fprintf...), altre cambiano leggermente. Restituiscono tutte EOF (End Of File).
- fscanf è utile quando si vuole trasferire il contenuto del file all'interno di una variabile da noi definita, che può essere un intero, un carattere o una stringa (file aperto in lettura).
- fprintf è utile quando vogliamo trasferire il contenuto di una nostra variabile sul file (file aperto in scrittura).