STRUTTURE DATI

Una struttura dati, o più semplicemente struttura, è un aggregato di dati omogeneo o eterogeneo che va a definire un nuovo "tipo" di dato composto da tipi già predefiniti.
Una strutta dati statica, come potrebbe essere un vettore oppure un tipo di dato per definire nome, sesso ed età di una persona, è un tipo di struttura che mantiene costante il numero di elementi presenti al suo interno durante l'esecuzione del programma. Capita talvolta di dover gestire strutture dati che richiedono un elevato numero di inserimenti o di cancellazioni, le strutture statiche possono risultare inadeguate a causa della loro scarsa flessibilità. Per evitare che dunque, durante l'esecuzione del programma, ci si accorga di aver allocato troppo o troppo poco spazio per le variabili da utilizzare si utilizzano le strutture dati dinamiche che ci permettono di occupare effettivamente solo la memoria che di volta in volta è necessaria.
I costituenti di base di queste strutture dovranno essere strutture statiche che verranno collegate fra loro attraverso l'utilizzo di una variabile puntatore. E' il modo in cui questi costituenti sono organizzati e collegati tra loro che renderà queste strutture effettivamente dinamiche. Le tre strutture dati dinamiche più importanti: la lista, la pila e la coda ma è la prima tra queste che ci permette una libertà di manipolazione dati maggiore.
STRUTTURA STATICA>>

```
typedf struct {
char nome[20];
char sesso;
int età;
}persona;
```

STRUTTURA DINAMICA>>

```
typedf struct rub{
char nome[20];
char numero[11];
struct rub *next;
}rubrica;
```

RICORSIONE E ITERAZIONE

Il metodo della ricorsione consiste nel suddividere il problema in problemi più semplici che una volta risolti andranno a costituire la soluzione del problema originario. In informatica una funzione è detta ricorsiva quando chiama se stessa, ovvero sia se all'interno di una funzione è presente una chiamata alla funzione stessa. Una funzione ricorsiva sa risolvere direttamente solo i casi particolarmente semplici di un problema, detti casi base; per questo motivo se la funzione viene chiamata passandole dati che costituiscono un caso base lo risolve e ritorna il risultato, altrimenti chiama se stessa passando un dato sempre più semplificato (ridotto) ogni qualvolta essa si richiama.
Ogni volta che la funzione chiama se stessa sospende la sua esecuzione per eseguire la nuova chiamata; il tutto termina quando la funzione più interna incontra un caso base.
Il linguaggio iterativo consiste invece nella sequenza di più istruzioni che vengono svolte l'una dopo l'altra senza sospensioni di esecuzione. Questo tipo di approccio è preferibile dal punto di vista dello spreco di memoria e della velocità di esecuzione rispetto a quello ricorsivo ma capita spesso che quest'ultimo sia più intuitivo rispetto al primo.
Si possono ottenere gli stessi risultati con un procedimento iterativo utilizzando cicli while, for e do-while.

FUNZIONE RICORSIVA>>

```
int fattoriale(int n) {
If (n==0 || n==1){
return 1;
}else{
return n*fattoriale(n-1);
}
```

FUNZIONE ITERATIVA>>

```
int fattoriale(int n){
while (n>0){
f=f*n;
n--;
}
return f;
}
```

LINGUAGGIO COMPILATO E INTERPRETATO

I linguaggi di programmazione si distinguono in linguaggi compilati e interpretati. Il primo, come il linguaggio C per esempio, si serve di un compilatore che traduce il codice sorgente, ovvero sia il codice di testo scritto dal programmatore, in linguaggio macchina e crea un file eseguibie composto da soli bit che può essere scritto in memoria ed eseguito. Questo tipo di linguaggio è sicuramente più lento perché ad ogni modifica deve essere chiamato il compilatore che deve costruire un nuovo file eseguibile ma limita in maniera esponenziale gli errori che possono essere commessi dal programmatore e che potrebbero bloccare il programma eseguibile in runtime. Questo tipo di linguaggi inoltre sono a tipizzazione forte ovvero sia costringono lo sviluppatore a dichiarare ogni variabile con il suo tipo e il suo nome prima di utilizzarla così da ridurre ancora gli errori che potrebbero essere commessi durante la scrittura.
Oltretutto i linguaggi compilati mettono a disposizione un tipo di file accessibile a tutti, l'eseguibile, che non ha bisogno di compilatori per essere avviato.
I linguaggi interpretati come MatLab invece, offrono una grandissima rapidità di modifica e di sviluppo del codice ma allo stesso tempo non assicurano il giusto funzionamento di quest'ultimo. I linguaggi interpretati, infatti, si servono di un "interprete" che ogni volta che viene scritta un'istruzione la codifica e la esegue immediatamente permettendo così di testare immediatamente il funzionamento del programma. Questo però fa sì che la memoria utilizzata dal programma sia nettamente maggiore e che per leggere il codice sorgente si abbia bisogno dello stesso interprete cosa talvolta scomoda se si vuole creare un programma accessibile da altri utenti. Matlab inoltre è un linguaggio a tipizzazione debole il che significa che permette di dichiarare le variabili semplicemente utilizzandole all'interno del programma. Questo, per quanto comodo, aumenta in maniera drastica la possibilità di errore.

LISTE, PILE E CODE

Liste, pile e code sono tre strutture dinamiche che consentono l'allocazione dinamica della memoria ovvero sia la collocazione di spazio per determinate variabili durante l'esecuzione del programma. Questo tipo di strutture consentono di creare programmi che non sprechino spazio in grado di allocare tanta memoria quanta serve.
Per lista si intende un insieme di elementi collegati tra loro da un rapporto di sequenzialità a cui si può accedere solo in maniera sequenziale (partendo dal primo elemento fino all'ultimo).
La coda è una lista gestita con il metodo FIFO (First In First Out). Un elemento può essere utilizzato e quindi, uscire dalla coda, solo se è il primo, cioè se si trova in testa alla coda. Quando deve essere inserito un nuovo elemento questo può essere immesso solo alla fine della coda.
La pila (stack) è una lista gestita con il metodo LIFO (Last In First Out). Sia l'estrazione di elementi sia l'inserimento avvengono solamente da un lato ed è importante gestire la testa della pila.

PUNTATORI

Quando dichiariamo una variabile a questa viene assegnato un indirizzo di cella, ovvero sia l'indirizzo della cella di memoria che è stato dedicato alla variabile; un puntatore è anch'esso una variabile che contiene l'indirizzo di cella della variabile a cui viene associato. Per definire un puntatore è necessario anteporre al nome della variabile un asterisco (*), chiamato operatore di deferenziazione.

L'importanza e l'utilità dei puntatori risiede nel fatto che si possono manipolare sia il puntatore che la variabile puntata (cioè la variabile memorizzata a quell'indirizzo di memoria) contemporaneamente e che talvolta, come per esempio nello scambio tra parametri attuali e formali nella chiamata ad un sottoprogramma, è più utile usare un puntatore che contiene l'indirizzo di memoria di una variabile piuttosto che la variabile stessa.

Nel caso, per esempio delle liste dinamiche, i puntatori sono fondamentali perché costituiscono il "collegamento" tra le varie celle di memoria della lista e ci permettono di spostare, eliminare, modificare e aggiungere celle semplicemente manipolando i puntatori e i loro indirizzi di memoria.

STRUTTURA DI UN CALCOLATORE – MEMORIA

Un calcolatore è costituito da una memoria principale, una memoria secondaria (o di masssa) e una CPU.
La memoria principale è suddivisa in due parti: la RAM e la ROM. La prima (Random Access Memory) è la memoria in cui vengono caricati i dati in maniera temporanea durante l'esecuzione dei programmi. Questo tipo di memoria è ad accesso casuale, il che significa che ogni dato è localizzato da delle coordinate che indentificano l'indirizzo di memoria, ed è volatile: si svuota ogni volta che il calcolatore viene spento.

L’altra parte della memoria principale, la ROM (Read Only Memory) è una memoria dalla quale è possibile solo leggere dati e non sovrascriverli. Questo tipo di memoria non è volatile e contiene i programmi per il corretto funzionamento del calcolatore come il BIOS.
L’altra parte fondamentale del calcolatore è la memoria di massa in cui è presente il disco rigido che si occupa di memorizzare in modo permanente dati che non riguardano l’esecuzione di programmi o altre funzionalità del calcolatore. Il disco rigido è costituito da materiale magnetico in cui i “bit” di informazione vengono memorizzati come punti magnetici che vengono incisi su di esso.
L’unità centrale di elaborazione (CPU) è contenuta nell’hardware e si occupa di effettuare operazione aritmetiche e logiche: può effettuare operazioni di somma, sottrazione, moltiplicazione e divisione sui dati e può prendere decisioni in base al risultato del confronto logico tra dati.