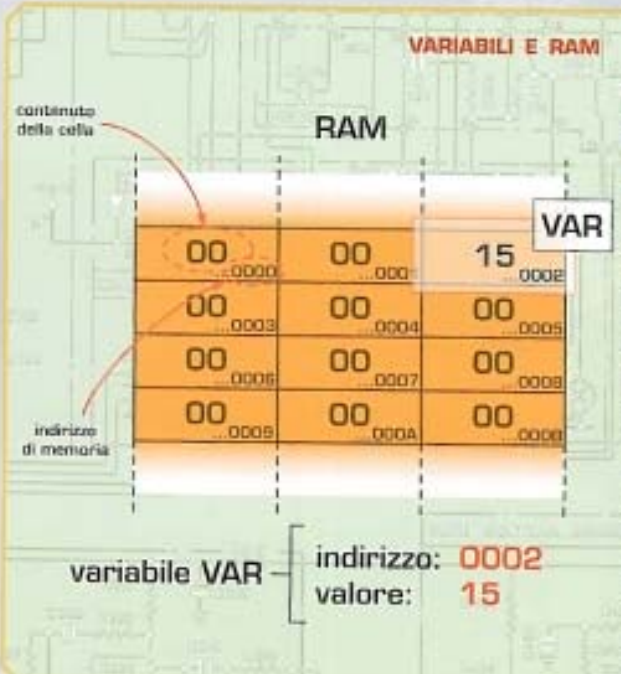
VARIABILI E TIPI DI DATI

In questo Workshop torniamo a parlare di programmazione esaminando il ruolo delle variabili all'interno del linguaggio C.

Nel fascicolo 16 abbiamo iniziato il nostro percorso tra le fondamenta del C, esaminando un semplicissimo esempio di programma. Facciamo ora un passo avanti e iniziamo a parlare di **variabili**. Per capire meglio di cosa si tratta, ripensiamo all'esempio che abbiamo citato nel fascicolo 16: la somma dei numeri da 0 a 9. Proprio in tale esempio siamo ricorsi all'uso di ciò che abbiamo definito **variabile di accumulazione** (o **accumulatore**) come **strumento necessario per memorizzare** le somme parziali all'interno del processo ciclico che risolveva il problema iniziale. Una variabile è proprio questo: una sorta di **contenitore che può essere impiegato per memorizzare e gestire dei dati.**

UN SALTO NELLA MEMORIA

Per capire meglio come 'funziona' una variabile, diamo un'occhiata da vicino alla struttura della **memoria volatile dei computer** (RAM, ***Random Access Memory*** ossia 'Memoria ad Accesso Casuale'). I comuni PC (ma potremmo affermare che questo discorso vale per tutti i sistemi dotati di memoria ad accesso casuale) organizzano la propria RAM come se fosse un'enorme griglia formata da milioni di **celle**, ognuna delle quali è identificata da un preciso **indirizzo numerico**. Accedere a una memoria RAM, quindi, significa avere a che fare con una rigorosa gestione degli indirizzi, che viene a complicarsi sempre di più con l'aumentare del numero di celle utilizzate. Le variabili risolvono questa problematica fornendo un sistema che permette di **associare nomi simbolici a precise aree di memoria. Il programmatore, quindi, non deve più preoccuparsi di 'dove' sono stati allocati i dati all'interno di questa griglia, ma può far semplicemente ricorso ai nomi che ha deciso di utilizzare**: tutto il processo di allocazione e risoluzione degli indirizzi viene gestito automaticamente dal software (il C offre, comunque, la possibilità di accedere



Le memorie ad accesso casuale (note come RAM, *Random Access Memory*) sono organizzate come un'enorme griglia di celle (il numero di celle presenti costituisce la dimensione della RAM), ognuna delle quali viene identificata da un preciso indirizzo. L'uso delle variabili (come la variabile 'VAR' mostrata nell'esempio a lato) consente di memorizzare e manipolare informazioni senza la necessità di preoccuparsi di dove esse siano fisicamente collocate all'interno di questa immensa griglia.

Una semplice rappresentazione di una porzione di RAM e dell'allocazione di una variabile 'VAR'.

direttamente alla memoria attraverso strumenti chiamati **puntatori**, che per ora non trattiamo).

VARIABILI DI OGNI TIPO

Abbiamo detto che una variabile è contraddistinta da un **nome simbolico**, ossia un nome che permette di attribuirle facilmente un riferimento mnemonico (il nome può essere scelto a piacimento dal programmatore, ma in generale si preferisce utilizzare nomi che richiamano 'il ruolo' della variabile all'interno del software). Abbiamo anche detto che ogni variabile rappresenta una precisa **'area di memoria'**, ma come viene gestito il contenuto di tali aree? Quanto saranno effettivamente 'grandi'? Tutto dipende dal **tipo** della variabile coinvolta. Il tipo di una variabile è un identificatore attraverso il qualeè possibile indicare al compilatore quale sarà il contenuto della variabile stessa, dimensionando, tra le altre cose, la quantità di memoria necessaria per la sua allocazione. Vediamo nella tabella sottostante cinque tipi di dati che utilizzeremo nel linguaggio C, con relativi significati e dimensioni in bit (8 bit equivalgono a 1 byte). Le variabili devono essere correttamente **tipizzate** in base al contesto di utilizzo: se ad esempio volessimo effettuare una **somma di numeri interi** potremmo far ricorso a una variabile di tipo **'int'**, mentre per **calcoli matematici e scientifici**, sarebbe meglio ricorrere a variabili **'float'** o **'double'**. Per il momento tralasciamo il significato del tipo **'void'**, a cui ricorreremo più avanti quando parleremo di **funzioni**.

DICHIARARE UNA VARIABILE

Per poter utilizzare una variabile è, innanzitutto, indispensabile **dichiararla. Il processo di dichiarazione all'interno di un programma C deve avvenire all'inizio dei blocchi di codice** (con l'avvento del C++ tale vincolo è stato abolito, ma noi faremo riferimento comunque alle regole del C originario). Dichiarare una variabile è molto semplice, è, infatti, sufficiente scrivere:

```
tipo nome_variabile
[= valore_iniziale];
```

In pratica si deve indicare come prima cosa il **tipo** della variabile, seguito dal **nome scelto**. Il nome può essere deciso a piacere, usando caratteri alfabetici e numerici, ma non deve iniziare con un numero; può, inoltre, includere il simbolo '_', ma non simboli e operatori matematici. Ad esempio è valido il nome 'variabile_1', mentre non è accettato il nome '1variabile'. Una variabile può, infine, essere inizializzata (ossia le può essere attribuito un valore iniziale) attraverso l'**operatore di assegnamento '='** (ad esempio **'int i=0;'** o **'char carattere='a';'**). Dalla sua dichiarazione, la variabile sarà utilizzabile all'interno del blocco di codice in cui è stata dichiarata (ma non all'esterno di esso! Vedi il box nella pagina successiva).

I MODIFICATORI DI TIPO

I tipi base che abbiamo appena visto possono essere 'modificati' utilizzando alcune parole chiave specifiche. La prima che trattiamo è il modificatore **'const'**, che consente di definire una **costante. Una costante, come intuibile dal nome, non è altro che una speciale variabile che non può essere modificata successivamente alla sua inizializzazione.** Se, ad esempio, volessimo definire la costante matematica 'Pi_Greco' (approssimata) da utilizzare all'interno del nostro programma, è sufficiente digitare l'istruzione:

```
const float Pi_Greco=
=3.14159;
```

Con questa riga di codice abbiamo creato una costante 'Pi_Greco' con valore 3,14159,

Tipo di dato	Significato	Dimensione
int	Numeri interi	32
char	Caratteri	8
float	Numeri decimali (virgola mobile)	32
double	Numeri decimali a doppia capacità (virgola mobile)	64
void	Tipo 'vuoto'	-

Una tabella essenziale che riassume i principali tipi di dati che utilizzeremo programmando in C.

Una variabile numerica di 32 bit nella quale il primo bit (ossia quello di 'peso maggiore') viene utilizzato come 'bit di segno'.

che non potrà più essere variato durante l'esecuzione del programma. Il secondo modificatore che vediamo è la parola chiave **'unsigned'**. Per comprenderne il significato è necessario fare un cenno alla rappresentazione dei numeri all'interno del calcolatore. Abbiamo già detto che i computer gestiscono i numeri attraverso sequenze binarie (ossia di 1 e 0), ma come è possibile capire se il numero memorizzato è positivo o negativo? Proprio utilizzando il primo di questi bit, che prende il nome di **'bit di segno'** (generalmente, se tale bit è pari a 1 il numero è considerato negativo). Vediamo cosa ne deriva. Consideriamo il tipo 'int' visto in precedenza: esso è formato da **32 bit, il primo dei quali ha funzione di segno dell'intero**. Ciò significa che **il numero vero e proprio viene codificato utilizzando 31 bit** dei 32 a disposizione. Una variabile 'int', quindi, potrà rappresentare ogni intero compreso tra $-(2^{31}-1)$ e $2^{31}-1$ (con 'n' bit, infatti, il numero più grande rappresentabile è pari a 2^n-1). La parola chiave **unsigned** non fa altro che 'rimuovere' il segno, **rendendo utilizzabili tutti i bit disponili**. Una variabile **unsigned int**, quindi, potrà rappresentare tutti gli interi da 0 a $2^{32}-1$. Gli ultimi due modificatori che vedremo in questo fascicolo sono le parole chiave **long** e **short**, che consentono di **variare la quantità di memoria riservata ai tipi**: **long** aumenta lo spazio fornendo un range numerico maggiore, **short** lo diminuisce. Ad esempio, sempre nel caso del tipo **int**, la dichiarazione **'short int'** riduce la dimensione della variabile **da 32 a 16 bit** ('**long int**', invece, non produce variazioni). Nel caso del tipo primitivo **double**, invece, la dicitura '**long double**' permette di **passare da 64 a 96 bit**. Logicamente è anche possibile **'combinare' i modificatori unsigned, long e short**. Nel prossimo Workshop dedicato alla programmazione, analizzeremo i principali operatori del C e concluderemo con alcuni esempi pratici.

LA VISIBILITÀ DELLE VARIABILI

Ogni variabile è sottoposta a precise regole di visibilità (in inglese *scope*) in base al punto in cui viene dichiarata. In particolare, ogni variabile è richiamabile e modificabile (in gergo si dice 'visibile') in tutti i sottoblocchi di codice del blocco in cui viene dichiarata. Per comprendere meglio il significato di quest'ultima affermazione, aiutiamoci con lo schema a lato. In esso è visualizzata la struttura di un programma C composto da alcuni sottoblocchi nidificati (ossia posti l'uno all'interno dell'altro), nei quali sono dichiarate alcune variabili. Le variabili dichiarate esternamente alla funzione main(), come Var1, sono dette variabili globali, in quanto sono visibili e modificabili da ogni punto del programma in questione. Le altre variabili, invece, sono locali e accessibili contestualmente alla posizione. Var2, ad esempio, è accessibile in tutti i sottoblocchi della funzione main (B1 e B2), mentre Var3 e Var4 lo sono solamente nel sottoblocco B2, e così via per tutte le variabili definite.