Il **linguaggio C** è sempre il primo che viene usato a scopo didattico da chi vuole imparare il funzionamento dei linguaggi di programmazione e del computer.
Questo perché il C è estremamente **simile** alle istruzioni in **linguaggio macchina** che il nostro computer esegue e questa sua vicinanza ci permette di **capire intuitivamente** le operazioni **ordinate** di **base** che il nostro computer deve eseguire per completare un programma, rimanendo pur sempre con una **sintassi comprensibile ad un lettore umano**.

Le informazioni e come vengono salvate

Prendiamo come esempio un semaforo che può avere fino a **4 stati diversi**:
0: spento
1: rosso
2: giallo
3: verde

Immaginiamo per un momento di voler inserire, in qualche modo, all'interno della memoria del nostro computer il valore giallo del nostro semaforo e che la nostra memoria sia come una grande tabella di numeri a **una cifra.**
Per mettere il valore del nostro semaforo scriveremo il suo numero all'interno della tabella.

/	/	2	/
/	/	/	/

Dove gli / rappresentano informazioni **sconosciute** che possono esserci o non.
Supponiamo per un momento di voler scrivere il valore 13.
Poiché 13 è un numero a **due cifre** avremo bisogno di un **nuovo spazio** nella nostra tabella

/	1	3	/
/	/	/	/

Ricordando che il computer ha **bisogno** di informazioni **codificate in binario** per **poterle processare**, avremo:
00: spento
01: rosso
10: giallo
11: verde
Quindi, per inserire giallo nella nostra memoria dovremo scrivere sia il numero 1 che il numero 0.

/	1	0	/
/	/	/	/

Ogni cella della nostra memoria, si chiama **bit** (binary digit), anche conosciuto come la **minima informazione memorizzabile**.
Il numero di bit **necessario** per codificare l'**informazione** di un semaforo è 2 (00, 01, 10, 11)
ATTENZIONE:
Il fatto che il valore sia 0, non altera il numero di bit utilizzato, saranno due bit di valore 0
Perché accade ciò?
Se scrivessimo solo uno 0 nella nostra tabella

/	/	0	/
/	/	/	/

E se poi avessimo bisogno di scrivere 10 in un successivo momento? Avremmo **bisogno** dello **spazio accanto**, **occupato** dallo /, andando a **sovrascrivere** quindi un'informazione che **potrebbe essere importante** per **altri programmi** del computer che in questo momento stanno **usando** quella zona di memoria.

Quindi, nel momento in cui abbiamo bisogno di **memorizzare** nella **memoria** l'i**nformazione** dello stato di un semaforo, **come facciamo?**
Nulla di più semplice.
Chiediamo al computer uno **spazio di due bit**.
Lui ci **trova** uno **spazio libero** (ovvero utilizzabile)e a questo punto possiamo scriverci liberamente qualsiasi informazione a **due bit**.
E questo vale per qualsiasi informazione, potremmo aver bisogno di due bit, ma anche, come vedremo di salvare
8 bit (1 byte)
Oppure 32 bit(4 byte)
O perché no
Anche 1024byte(un kilobyte o 8kilobit)
Basta chiedere e ci sarà dato, ma l'importante è:
se **dichiariamo** di aver bisogno di una **certa quantità** di memoria, **non dobbiamo e non possiamo** usarne di più.
Il C (e il computer) non lo permettono.

Le variabili

Queste **quantità di spazi di memoria**, in C, sono **predefiniti**:
Se abbiamo bisogno di memorizzare un **numero intero** esso verrà salvato in uno spazio di memoria a **32bit(4byte)**, dove il **primo bit indica il segno** questo vuol dire che il numero più grande che possiamo memorizzare in un intero è:
0 1 1 1 1 1 1 1_1 1 1 1 1 1 1 1_1 1 1 1 1 1 1 1_ 1 1 1 1 1 1 1 1
8bit 8bit 8bit 8bit = 32bit
Se trasformiamo questo numero in binario con la regoletta che ci dice:
sommare ogni cifra per due elevato alla posizione da destra verso sinistra
(esempio 100=$0*2^0 + 0*2^1 + 1*2^2$ quindi 4)
Ci rendiamo conto che il numero **massimo più grande** codificabile con **32bit**, dove ricordiamo che il **primo indica il segno**, è:
2147483647(2^{32}-1)
Di conseguenza, grazie al primo bit, il numero più piccolo codificabile è :
-2147483647(-2^{32}+1)
In realtà per alcuni motivi di conversione che non spiego è -2147483648,
quindi in totale sono esprimibili 2^{32} tra numeri positivi e negativi, zero compreso.

In C, per **chiedere** al nostro computer uno **spazio di memoria** grande un intero non dobbiamo far altro che scrivere:
int var;
Con questa pratica riga tutto ciò che stiamo facendo è dire
"Ehi computer, mi servono 32 bit per scriverci dentro un intero"

Il computer **trova 32bit** di memoria **libera** (ricordiamo, 32 riquadri della tabella) e mi dice "Ok, eccoti i 32 bit, per riferti ad essi puoi chiamarli var"
A questo punto possiamo usare i nostri bit come ci pare e piace, **purché dentro ci mettiamo interi**, positivi o negativi che siano, non fa differenza.
Vogliamo scriveci dentro -123321?
Nulla di più semplice e intuitivo:
var = -123321;
In memoria comparirà
11111111111111100001111001000111
Dove ogni cifra sarà un bit.
Abbiamo appena imparato a memorizzare un'informazione nel nostro computer.