Un elemento fondamentale per un qualsiasi tipo di comunicazione è un alfabeto che ci permette di esprimere le informazioni che stiamo trattando in un linguaggio comprensibile.

Negli elaboratori, e più generalmente in tutti i sistemi digitali, si usa un alfabeto costituito semplicemente da due numeri: 0 e 1, chiamati bit. 8 bit vengono chiamati 1 byte. La scelta di questo tipo di alfabeto si può ricondurre al fatto che:

* I calcoli svolti in binario sono più veloci rispetto ad altre base di numerazione
* Costruire un dispositivo che sia in grado di gestire due soli stati è più semplice e meno costoso
* E’ più affidabile in quanto essendo solo due gli stati possibili delle grandezze fisiche è più semplice individuare eventuali errori

Anche le istruzioni e i dati dei nostri programmi, dopo opportune conversioni da parte del compilatore, sono tradotti in codice binario in modo che siano comprensibili alla macchina. Ogni tipo di dato e istruzione occupa un certo numero di bit (ovvero 0 o 1) nella memoria.

Come in ogni linguaggio, anche nel codice binario sono stati definiti gli standard per i quali è possibile associare ad una particolare sequenza di bit un certo valore. Tra questi il più importante è il codice ASCII che utilizzando 7 bit ci permette di rappresentare un massimo di 27 = 128 caratteri (52 tra lettere maiuscolo e minuscole, 10 cifre, 30 segni d’interpunzione e 30 caratteri controllo).

Lo spazio che verrà riservato in memoria per memorizzare i dati dipende dalla scelta che viene fatta dal programmatore durante la dichiarazione dei tipi di dato. Per quanto riguarda la costanti, esse in C sono trattate direttamente dal preprocessore che si occupa di cercare la costante introdotta e sostituirla con il valore definito durante la dichiarazione della costante stessa. Per questo una costante in C non riserva un proprio spazio in memoria, ma è per lo più un “cerca e sostituisci” che viene gestito dal preprocessore.

Stessa cosa non si può dire per le variabili o le strutture dei dati in quanto esse nel momento in cui vengono dichiarate riservano uno spazio in memoria: per esempio nel caso delle variabili un intero riserva 2 o 4 byte a seconda dell’architettura dell’elaboratore, un tipo double riserva 8 byte e un char ne riserva solo uno.

Inoltre sulla rappresentazioni degli interi si potrebbe aprire un discorso molto ampio in quanto esistono vari “metodi” per convertire un numero da decimale a binario e viceversa, specialmente se si usano dei numeri negativi. E’ importante sapere che la rappresentazione più usata è quella del complemento a due che è quella più veloce e consente di eseguire dei controlli per eventuali errori di overflow.

Per le strutture di dati la situazione è più complessa, per esempio in un array composto da 5 elementi di tipo intero, il sistemo operativo dovrà occuparsi di riservare 20 byte (5\*4) consecutivi nella memoria RAM.

Se inoltre si vogliono svolgere delle operazioni sui dati attraverso delle istruzioni, il sistema operativo dovrà occuparsi di riservare dello spazio in memoria non solo per l’istruzione stessa ma anche i dati da mettere in relazione. Per esempio utilizzando un’istruzione che ci permette di eseguire la somma, oltre a riservare dello spazio per l’istruzione stessa è necessario che sia specificato (anche implicitamente) dove leggere i due operandi da sommare e dove scrivere il risultato.

Fisicamente, tutti i dati e le istruzioni di cui abbiamo parlato fino adesso vengono salvati nella memoria volatile RAM che è in grado di conservare i dati solamente per il tempo in cui sono necessari e vengono persi immediatamente alla chiusura del programma o allo spegnimento del computer. Se si vuole evitare questo problema, è necessario salvarli in una memoria secondaria come un Hard Disk o un DVD.