7 : I DATI E LA LORO STRUTTURA NELLA PROGRAMMAZIONE

TIPO DI DATO

Un **tipo di dato** è una entità caratterizzata dai seguenti elementi:

- **un insieme X di valori** che raprresenta il "*dominio*" del tipo di dato;
- **un insieme non vuoto di costanti** che caratterizzano l'insieme X;
- **un insieme di operazioni** che si possono effettuare sull'insieme X.

I tipi di dato possono essere classificati in :

a) **tipi elementari o tipi semplici**, i cui dati non sono costituiti da altri dati;

b) **tipi strutturati,** i cui dati sono aggregazioni di tipi elementari che è possibile estrarrre tramite opportune operazioni.

Ogni linguaggio di programmazione prevede alcuni tipi di dato. **Qui tratteremo i più usati**.
La maggior parte dei linguaggi di programmazione consente al programmatore di definire propri tipi di dato così da renderli più vicini al problema in esame.
Questi tipi di dati sono conosciuti come **TIPI DI DATI ASTRATTI o ADT Abstract Data Type** e vanno trattati a parte.

In pratica attraverso la tecnica dell'ADT è possibile definire nuovi tipi di dato sostanzialmente più complessi e maggiormente inerenti la realtà del problema osservato. Si parla quindi di **astrazione sui dati** in quanto si realizza una astrazione sia dalla loro realizzazione fisica sia dalla loro implementazione dettagliando il nuovo dato in funzione delle operazioni ammesse per la loro manipolazione.

In altre parole durante l'attività di programmazione nel passaggio dal **problema** al **programma** eseguibile finale il concetto di **modello dei dati** o **struttura dati** compare trasversalmente:

(1) **a livello del progetto** ossia durante la fase astratta indipendentemente dalla tecnologia

(2) **a livello di codifica** ossia durante la fase di codifica o implementazione del programma

(3) **a livello interno o fisico** ossia a livello della modalità di memorizzazione a livello demìlla memoria perincipale e/o secondarire dell'elaboratore.

1) A LIVELLO DEL PROGETTO **con il termine "struttura dati"** si intende una STRUTTURA DATI ASTRATTA che è definita da:

- **un insieme base di elementi** in genere variabili che contengono i dati del problema
- **un insieme di regole** che determinano le relazioni tra i singoli elementi dell'insieme base;
- **un elenco di operazioni** che agiscono sull'insieme**.**

Invece di "struttura dati astratta" si utilizzano anche i termini "**modello dei dati"** o "**tipo di dato astratto o ADT".**

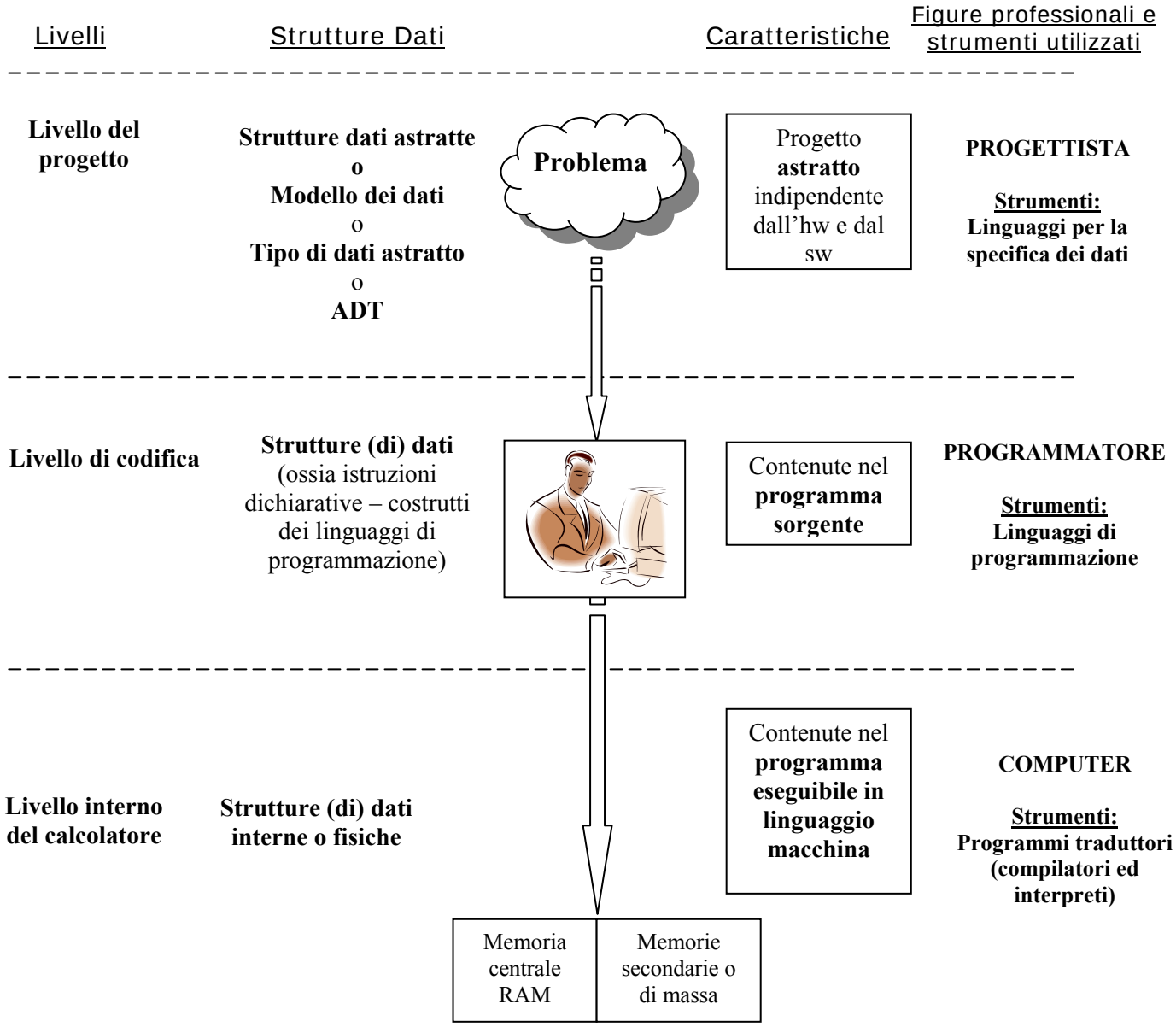
Il modello dei dati:

- NON dipende dal computer usato (hardware);
- NON dipende dal linguaggio di programmazione scelto per tradurre la struttura dati astratta e l'algoritmo nel programma sorgente.

Quindi il vantaggio di fornire una soluzione astratta del problema permette di NON ridefinire da capo il modello dei dati e l'algoritmo risolutivo ogni qualvolta si cambia elaboratore e/o linguaggio di programmazione.

2) A LIVELLO DI CODIFICA O IMPLEMENTAZIONE **con il termine "struttura dati"** si intendono i costrutti dei linguaggi di programmazione impiegati per realizzare nel programma sorgente il modello dei dati ossia la struttura dati astratta.
I **costrutti** dei linguaggi di programmazione per la **definizione delle strutture dati** si presentano come opportune frasi dichiarative, quindi non eseguibili, che descrivono al programma traduttore (compilatore o interprete) come organizzare i dati nel programma sorgente.

2) A LIVELLO INTERNO O FISICO **con il termine "struttura dati"** si intendono quelle strutture "fisiche" impiegate per memorizzare i dati nella memoria principale e/o di massa di un computer ossia sequenze di bit memorizzate e codificate opportunamente.



TIPO DI DATI INTERO

Osservazione importante iniziale

Quando parliamo di insieme dei numeri interi (relativi) **in programmazione** ci riferiremo sempre in realtà ad un suo sottoinsieme. Infatti interno di un computer pur aumentando a dismisura il numero di bit (o di byte) impiegati per rappresentare un intero e detto **N** il più grande intero rappresentabile, esisterebbe comunque un intero **N + 1** non rappresentabile.
Stesso discorso vale per il numero $\mathbf{N^1}$ ossia il numero più piccolo degli interi rappresentabili per il quale esisterebbe comunque un intero non rappresentabile $\mathbf{N^1 - 1}$.
Quindi in realtà il sottoinsieme dei numeri interi (relativi) rappresentato all'interno di un computer è un sottoinsieme limitato sia supoeriormente (da **N + 1**) sia inferiormente (da $\mathbf{N^1 - 1}$)

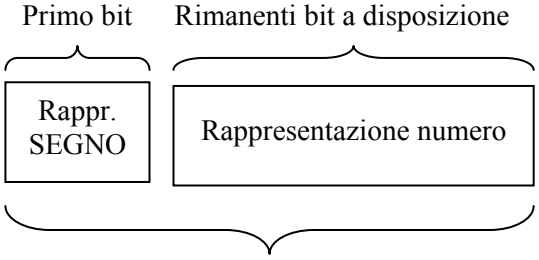
Più in dettaglio analizziamo alcuni modelli ri rappresentazione "fisica" usata dagli elaboratori per rappresentare il tipo di dato **intero** nei seguenti casi:

a) I NUMERI INTERI (ossia i naturali N): è importante per ciascuno schema di rappresentazione seguire modelli che indicano quanto spazio deve essere utilizzato per la rappresentazione di ciascun numero e quale il significato di ogni bit

TIPO	PAROLA DI MEMORIA	RANGE DI RAPPRESENTAZIONE
Byte senza segno	1 byte = 8 bit	$0 \leq n \leq 255$ (ossia $2^8 - 1$)
Parola senza segno	2 byte = 16 bit	$0 \leq n \leq 65535$ (ossia $2^{16} - 1$)
Doppia Parola senza segno	4 byte = 32 bit	$0 \leq n \leq 4{,}3*10^9$ (ossia $2^{32} - 1$)

b) I NUMERI INTERI CON SEGNO (ossia i gli interi relativi Z): non si hanno ancora i problemi connessi alla rapprsentazione dei numeri contenenti la virgola, ma c'è il problema della rappresentazione del segno + o – e del numero.

In genere **il primo modello** utilizzato per la rappresentazione prevede la seguente ripartizione tra segno e numero n base ai bit totali a disposizione:



con il primo bit destinato alla rappresentazione del segno valorizzato

- con **0** se il numero è *positivo*
- con **1** se il numero è *negativo*

e con le cifre binarie rappresentative il numero allineate a partire da destra con gli eventuali bit non utilizzati posti a zero perché non significativi.

In base al numero di bit messo a disposizione per la rappresentazione dei numeri interi relativi avrò a disposizione intervalli differenti di valori:

TIPO	PAROLA DI MEMORIA	RANGE DI RAPPRESENTAZIONE
Byte relativo	1 byte = 8 bit	$-128 \leq n \leq 127$ ossia $[-2^7 ; -2^7 - 1]$
Parola relativo	2 byte = 16 bit	$-32768 \leq n \leq +32767$ ossia $[-2^{15} ; -2^{15} - 1]$
Doppia Parola relativo	4 byte = 32 bit	$-2^{31} \leq n \leq -2^{31} - 1$

N.B. La diferenza di una unità tra il totale dei numeri relativi negativi rappresentati ed il totale dei numeri relativi positivi rappresentati è dovuta al fatto che lo **zero** 0 non essendo né positivo né negativo **lo si considera come positivo** e quindi abbassa di una unità i numeri positivi rappresentabili

Il secondo modello (metodo più utilizzato) seguito per effettuare tale rappresentazione è quella detta **per complemento alla base** (dove la base in questo caso è pari a 2)
Tale modello si basa sulla seguente definizione di **complemento alla base del numero x nella base p**:

$$\textbf{Rappr(x)} = \begin{cases} \mathbf{x} & \textbf{se } \mathbf{x \geq 0} \\ \mathbf{2^p - |\, x\,|} & \textbf{se } \mathbf{x < 0} \end{cases}$$

dove **p è** il numero di bit a disposizione per rappresentare il numero in binario.
Il range di variabilità ammessa va dalla **parte intera alta di $-(2^p - 1) / 2$** indicata con **PA** alla **parte intera bassa di $(2^p - 1) / 2$** indicata con **PB** ossia

$$\mathbf{PA[-(2^p - 1) / 2] \, , \, PB[(2^p - 1) / 2]}$$

TIPO	PAROLA DI MEMORIA	RANGE DI RAPPRESENTAZIONE
Byte relativo	1 byte = 8 bit	$-128 \leq n \leq +127$ ossia $PA[-(2^8 - 1)/2], PB[(2^8 - 1)/2]$
Parola relativo	2 byte = 16 bit	$-32768 \leq n \leq +32767$ ossia $PA[-(2^{16} - 1)/2], PB[(2^{16} - 1)/2]$
Doppia Parola relativo	4 byte = 32 bit	$PA[-(2^{32} - 1)/2] \leq n \leq PB[(2^{32} - 1)/2]$

N.B. Il vantaggio di un tale tipo di rappresentazione permette di effettuare l'operazione di addizione in modo indipendente dal segno degli operandi usando lo stesso algoritmo usato sui numeri naturali a patto di troncare il risultato alle p cifre disponibili

Il terzo modello viene usato per la rappresentazione di numeri che possono essere convertiti rapidamente in cifre decimali ed e meglio noto con il nome **codice BCD o Binary Code Decimal.**
In ogni byte vengono rappresentate due cifre decimali, ogni cifra del numero decimale occuoa 4 bit che assumono il valore della corrispondente rappresentazione binaria.
Sappiamo che con 4 bit le configuraioni possibili sono 16 ma per rappresentare le cifre da 0 a 9 ne vengono usate solo 10 lasciando 6 inutilizzate (codice ridondante).

Operatori aritmetici associati al tipo intero

+ addizione
- sottrazione
* moltiplicazione
DIV divisione intera (+ DIV + fa +; + DIV - fa -;- DIV + fa -; - DIV - fa +)
MOD resto della divisione tra gli operatori indicati (solo se secondo operando positivo, altrimenti errore)

Operatori relazionali associati al tipo intero

< minore
≤ minore o uguale
> maggiore
≥ maggiore o uguale
= uguale
≠ diverso

Tali operatori forniscono un risultato boolenao ossia che assume valore solo VERO oppure solo FALSO.

E' possibile avere sia **errori di overflow** (ossia quando il risultato risulta più grande del massimo numero rappresentabile sia **errori di underflow** (ossia quando il risultato risulta più piccolo del minimo numero rappresentabile).
L'errore di divisione per 0 causa **errore di run-time** e si verifica per un superamento della capacità della memoria nel ricevere un valore che supera i limiti previsti.

N.B. Nella PSEUDOCODIFICA per la dichiarazione di una variabile di questo tipo usare **INT**

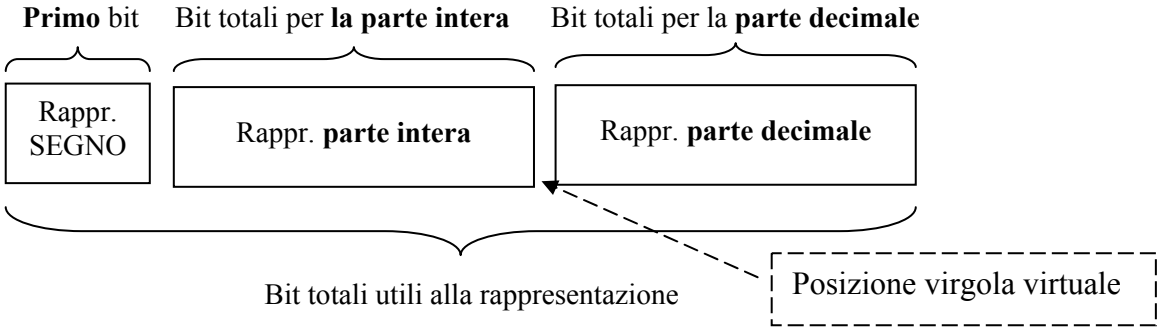
TIPO DI DATI REALE

Osservazione importante iniziale

Come per gli interi (relativi) quando parliamo di insieme dei numeri reali **in programmazione** ci riferiremo sempre in realtà ad un suo sottoinsieme. Inoltre a causa delle approssimazioni sia per eccesso che per difetto può accdere che relazioni perfettamente valide in matematica non lo siano una volta eseguite al computer (Esempio 1/3 + 1/3 + 1/3 = 1).

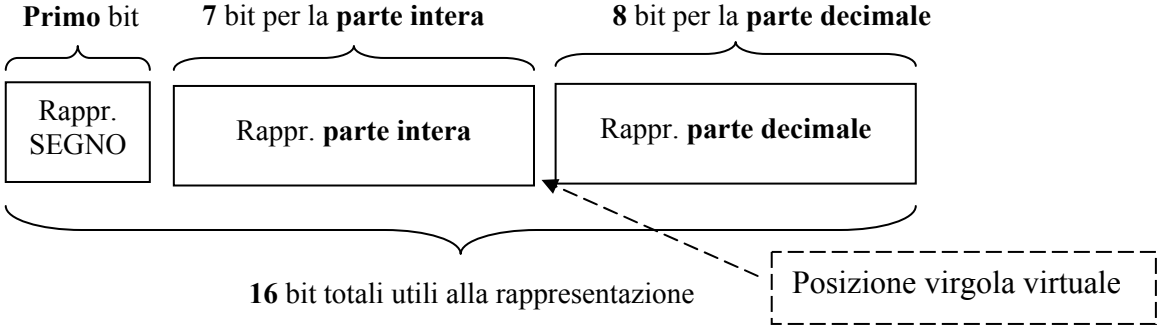
Più in dettaglio analizziamo alcuni modelli ri rappresentazione "fisica" usata dagli elaboratori per rappresentare il tipo di dato **reale** (numeri con virgola)

a) Rappresentazione a virgola fissa o FIXED POINT

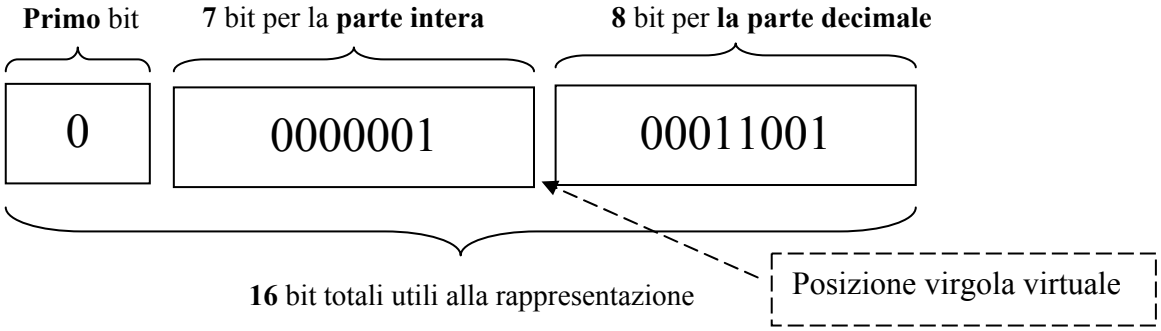


Viene chiamato così perché il numero di posti per rappresentare le cifre dopo la virgola sono prefissati ossia la virgola occupa sempre lo stesso posto *virtuale*

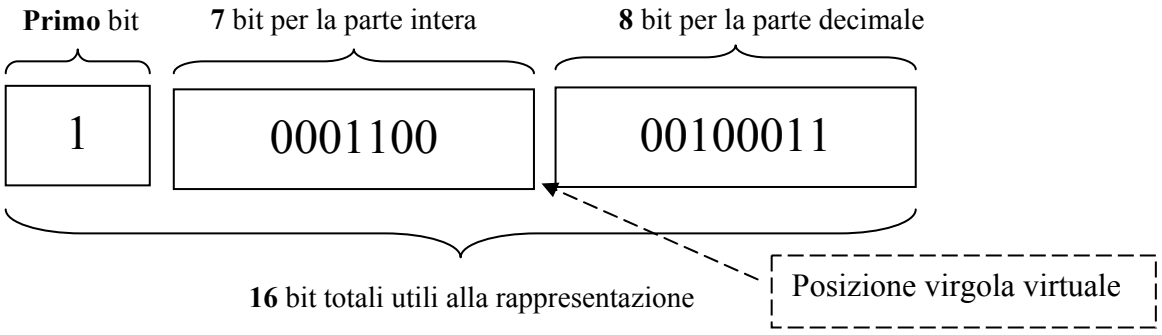
Esempio: Supponiamo una rappresentazione a virgola fissa basata su 16 bit ossia che segue il seguente modello



In questo modello di rappresentazione sia la parte intera che la parte decimale vengono convertite come numeri indipendenti secondo le regole di conversione già illustrate
In questo caso il numero **+1,25** verrà rappresentato come segue



In questo caso il numero **-12,35** verrà rappresentato come segue



TIPO	PAROLA DI MEMORIA	RANGE DI RAPPRESENTAZIONE
Parola virgola fissa	2 byte = 16 bit	Nelle ipotesi fatte il numero minimo rappresentabile è $N_{min} = -(2^7 - 1), (2^8 - 1)$ Mentre il numero massimo rappresentabile è $N_{Max} = +(2^7 - 1), (2^8 - 1)$
Doppia Parola virgola fissa	4 byte = 32 bit	Nelle ipotesi fatte il numero minimo rappresentabile è $N_{min} = -(2^{15} - 1), (2^{16} - 1)$ Mentre il numero massimo rappresentabile è $N_{Max} = +(2^{15} - 1), (2^{16} - 1)$

b) Rappresentazione a virgola mobile o FLOATING POINT (<u>secondo lo standard IEEE 754</u>)

E' diventata il modello di rappresentazione per i numeri relai più usato dai costruttori di computer. L'idea alla base di tael modello è quella di far variare dinamicamente (in inglese *floating* significa fluttuante) la posizione della virgola (in inglese *point* signifiva virgola).
In questo caso il numero viene espresso secondo *la notazione scientifica* ossia diviso in due parti dette **mantissa** ed **esponente** (in questo caso essendo la base uguale a 2 si chiama **caratteristica**)

In questo modello di rappresentazione:

- **la mantissa** normalizzata (ossia con la virgola immediatamente a sinistra della prima cifra diversa da zero da rappresentare) rappresenta un numero compreso tra 0,0000…1 a 0,1111…1
- **la caratteristica** indica la potenza di **2** per cui occorre moltiplicare la mantissa al fine di ottenere il numero da rappresentare.

+/- , <mantissa *normalizzata*> * 2 +/- <caratteristica>

Quindi per rappresentare un numero reale in virgola mobile con il computer occorre stabilire un certo numero **p** di bit per la **mantissa** ed un certo numero **q** di bit per **l'esponente o caratteristica**.

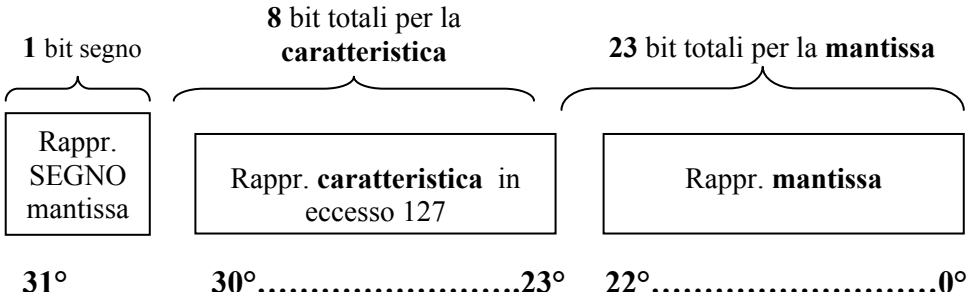
La convenzione più usata è quella IEEE754 che prevede:

1. la mantissa rappresentata in **modulo** e **segno**;
2. la caartteristica rappresentata in **eccesso 127 in precisone singola** per evitare l'uso del segno

N.B. La caratteristica è rappresentata in eccesso 127 in precisone singola se:

- agli esponenti positivi viene aggiunto 127 all'esponente;
- agli esponenti negativi viene sottatto 127 all'esponente;

Nella precisone singola abbiamo **32 bit** a disposizione per la rappresentazione dei numeri reali a virgola mobile e quindi il modello rappresentativo può essere così schematizzato



32 bit totali utili alla rappresentazione

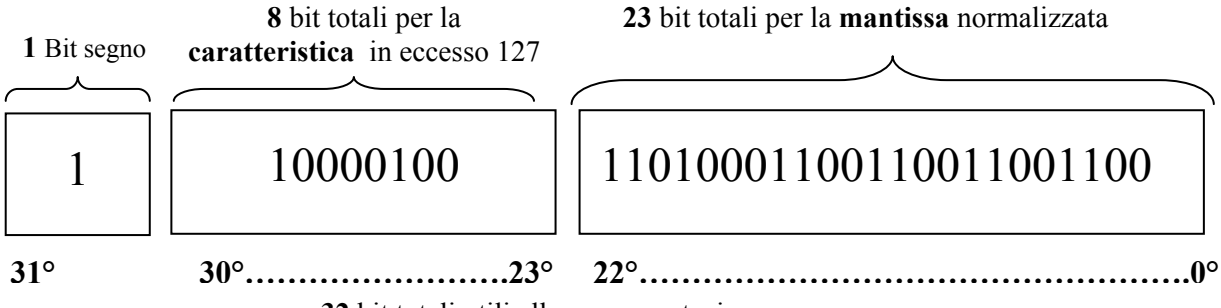
Supponiamo di volere rappresentare con questo modello il numero **-13,10** utilizzando **23** cifre per la mantissa normalizzata ed **8** per la caratteristica in eccesso 127

- 13, 10 = - 1101 , 0 0011 0011…..

normalizzando il numero avrò **– 0,1101 00011 * 2** ^{0100}
il numero è **periodico** e quindi occorrerà troncare le cifre della mantissa ed è **negativo**
inoltre l'esponente **0100** (4 in decimale) **in eccesso 127** è

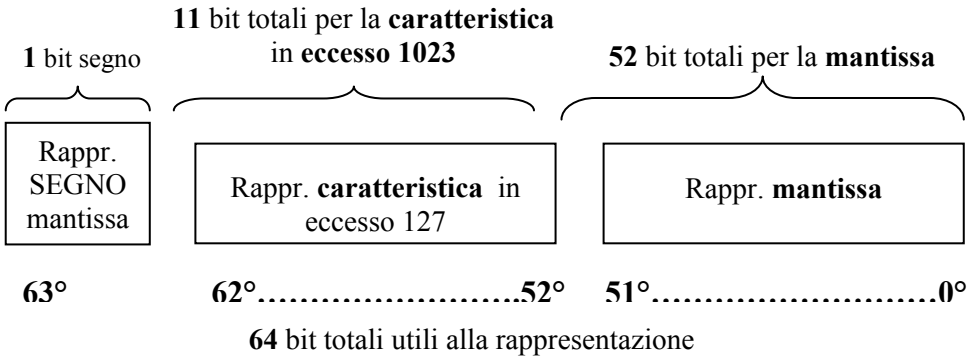
01111111 + 00000100 = **10000100**
(ossia in decimale 127 + 4 = 131)

quindi la rappresentazione in virgola mobile in singola precisione di **-13,10** sarà



32 bit totali utili alla rappresentazione

Nella precisone doppia abbiamo **64 bit** a disposizione per la rappresentazione dei numeri reali a virgola mobile e quindi il modello rappresentativo può essere così schematizzato



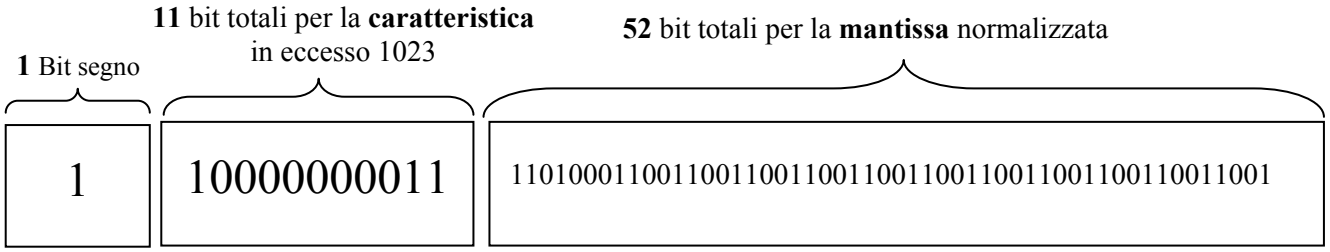
Supponiamo di volere rappresentare con questo modello sempre il numero **-13,10** utilizzando **52** cifre per la mantissa normalizzata ed **11** per la caratteristica in eccesso 1023

- 13, 10 = - 1101 , 0 0011 0011…..

normalizzando il numero avrò **– 0,1101 00011 * 2** ^{0100}
il numero è **periodico** e quindi occorrerà troncare le cifre della mantissa ed è **negativo**
inoltre l'esponente **0100** (4 in decimale) **in eccesso 1023** è

01111111111 + 00000000100 = **10000000011**
(ossia in decimale 1023 + 4 = 1027)

quindi la rappresentazione in virgola mobile in doppia precisione di **-13,10** sarà



Operatori aritmetici associati al tipo reale

+	addizione
-	sottrazione
*	moltiplicazione
/	divisione (+ / + fa +; + / - fa -;- / + fa -; - / - fa +)

Operatori relazionali associati al tipo reale

<	minore
≤	minore o uguale
>	maggiore
≥	maggiore o uguale
=	uguale
≠	diverso

Tali operatori forniscono un risultato boolenao ossia che assume valore solo VERO oppure solo FALSO.

E' possibile avere sia **errori di overflow** (ossia quando il risultato risulta più grande del massimo numero rappresentabile sia **errori di underflow** (ossia quando il risultato risulta più piccolo del minimo numero rappresentabile).
L'errore di divisione per 0 causa **errore di run-time** e si verifica per un superamento della capacità della memoria nel ricevere un valore che supera i limiti previsti.

N.B. Per la PSEUDOCODIFICA: REAL (singola pecisione o doppia)

TIPO DI DATI CARATTERE

Lo standard **ISO 646** (ISO = International Standards Organization) comprende 128 caratteri di base ossia:

- 26 lettere MAIUSCOLE A,B;C,……..Z
- 26 lettere minuscole a,b;c,……..z
- le 10 cifre numeriche 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- lo spazio o *blank* (indicato con ␣)
- i segni di interpunzione ; , . ' ? ! etc.
- molti altri simboli non compresi sulla tastiera del computer ma ugualmente rappresentabili

Per rappresentare questo alfabeto all'interno della macchina che utilizza un alfabeto molto più spartano costituito da sue soli simboli **0** ed **1** (alfabeto binario), occorre stabilire una corrispondenza biunivoca tra due alfabeti ossia occorre che ad ogni simbolo del primo albabeto corrisponda una ed una sola sequenza di simboli del secondo alfabeto: tale corrispondenza tra insieme di caratteri da rappresentare e parole di un codice prende il nome di **codifica.**
I codici si differenzianoper il numero di bit che usano per la rappresentazione di ogni carattere.
Per definire un **set di caratteri** occorre specificare:

- il set di codici utilizzato
- il set di caratteri rappresentato
- la corrispondenza tra questi

Il set di caratteri **ISO 646** usa la codifica **ASCII** (American Standard Code for Information Interchange) a **7 bit** o US ASCII (detto anche ASCII di base o ASCII ristretto): tale codifica usa 8 bit per carettere ma lascia inutilizzato il bit più significativo (ossia il primo a partire da sinistra) ponendolo sempre a 0. Quindi in pratica usa 7 bit che permettono di ottenere appunto $\mathbf{2^7 = 128}$ combinazioni diverse.
Ad ogni carattere viene associato in modo convenzionale un numero nella sua rappresentazione binaria compreso tra 0 e 127.

I primi 32 caratteri (sequenze) del codice ASCII contengono quelli che vengono chiamati **i caratteri di controllo** che non fanno parte dell'alfabeto stampabile.
Le **cifre** da 0 a 9 hanno tutte lo stesso valore (0011) nel primo semibyte quello più significativo, mentre il secondo semibyte contiene la rappresentazione della cifra in binario assoluto.
Le **lettere** sono suddivise in maiuscole e minuscole; in ognuno dei due gruppi è stato mantenuto l'ordine alfabetico. Sono possibili operazioni di confronto ed ordinamento su caratteri basandosi sul loro codice ASCII e se si considerano omogeneamente (ossia solo sulle minuscole o solo sulle maiuscole). Infatti considerato in senso più ampio avremmo che **B < a** e l'ordinamento risulta alterato.

Di fatto soltanto per l'inglese, il latino e la lingua swahili sono sufficienti 128 caratteri di base: tutte le altre lingue hanno bisogno di altri caratteri (ad esempio anche l'italiano usa le lettere acecentate non presenti tra i caratteri di base).
Il bit più significativo degli 8 bit del codice ASCII di base o risatretto può ewssere usato per aggiungere di fatto altri 128 caratteri poiche con 8 bit a disposizione è possibile rappresentare $\mathbf{2^8 = 256}$ combinazioni diverse tra loro.

NON ESISTE UN UNICO CODICE ASCII AD 8 BIT ma ne esistono diversi **chiamati tabelle di codici** a seconda della lingua da trattere che usa il proprio alfabeto.
TUTTI I CODICI ASCII AD 8 BIT SONO COMPATIBILI CON IL CODICE ASCII A 7 BIT (ossia hanno i primi 128 caratteri rappresentati uguali).
L'ISO ha ovviamente regolamentato tutte queste estensioni con la serie **ISO 8859:** quello 8859-1 chimato anche **ISO Latin-1** contiene i caratteri dell'alfabeto latino più esteso in grado di rappresentare la maggioranza delle lingue europee (italiano compreso) ed è stato riconosciuto come **ASCII esteso ad 8 bit.**

Il codice ASCII ad 8 bit non è adatto alle lingue orientali che usano migliaia di ideogrammi ciascuno dei quali corrisponde **non** ad un suono (come invece fa una lettera) ma ad un intero concetto o parola. E' necessaria una codifica che utilizzi almeno 16 bit e quindi $\mathbf{2^{16} = 65536}$ possibili caratteri.

Nel 1991 è stato istituito il consorzio Unicode che ha definito uno standard a 16 bit noto con il nome di set di caratteri **UNICODE** che comprende le lettere di tutti gli alfabeti, molti simboli speciali ed ideogrammi; se un elemento appartiene a più lingue compare una volta sola e questo vale anche per l'insieme degli ideogrammi indicato con le lettere CJK (cinese, giapponese, coreano.
I primi 128 caratteri sono gli stessi del codice ASCII ristretto ed i primi 256 sono gli stessi del codice ASCII ad 8 bit esteso o Latin-1.

Nel 1993 l'ISO ha definito un nuovo standard chiamato **ISO 10646** e la relativa codifica **UCS-4** ossia Universal Character Set a 4 byte.
Per la codifica vengono usati **4 byte** con il primo bit posto a 0 e quindi con effettivamente **31 bit** a disposizione che permettono di rappresentare $\mathbf{2^{31} = 2.147.483.648}$ caratteri.
In questo standard un blocco di **256 posizioni** di codifica successive viene chiamato **riga**, **256 righe** costituiscono un **livello** e **256 livelli** un **gruppo**. In tutto ci sono 128 gruppi.
Questo set di carettri 10646 è il set di caratteri indicato nella specifica del lòinguaggio HTML 4.0

Il **dato** di **tipo carattere** va racchiuso tra apici singoli (ciò è fondamentale per distinguere il numero **9** dalla carattere **'9'**). Una **stringa** è un insieme di caratteri che, come il singolo carattere, va racchiusa tra apici.
Operatori relazionali associati al tipo carattere

$<$	minore
$\leq$	minore o uguale
$>$	maggiore
$\geq$	maggiore o uguale
$=$	uguale
$\neq$	diverso

Tali operatori forniscono un risultato boolenao ossia che assume valore solo VERO oppure solo FALSO e vengono risolti in base al valore relativo asociato contenuto nel codice ASCII.

N.B. Nella PSEUDOCODIFICA per la dichiarazione di una variabile di questo tipo usare **REAL**

TIPO DI DATI BOOLEANO

Tale tipo di dato risulta fondamentale nei costrutti di controllo che prevedono la specificazione di condizioni attraverso l'algebra proposizionale o di Boole.
Tale tipo di dato può assumere soltanto due costanti logiche (valori di verità) che vengono dette VERO e FALSO. Tali due valori risultano ordinati all'interno dell'insieme con FALSO < VERO.
Gli operatori che possono essere applicati a questi dati sono i connettivi logici:

AND	congiunzione logica
OR	disgiunzione logica
NOT	negazione logica

e forniscono risultati boolenani in accordo alle relative tabelle di verità definite per ciascuno di essi.
In questo contesto si assume la precedenza dell'operatore NOT sull'operatore AND e dell'operatore AND sull'operatore OR.

N.B. Nella PSEUDOCODIFICA per la dichiarazione di una variabile di questo tipo usare **BOOL**