Introduzione alla programmazione in linguaggio C

Il primo programma in C

```
/* PRIMO - Il primo programma C */


#include <stdio.h>

main()
{
    printf("Io so programmare in C!");
}
```

commento

Header della libreria

Funzione principale

Ogni istruzione in C va terminata con un ;

Funzioni

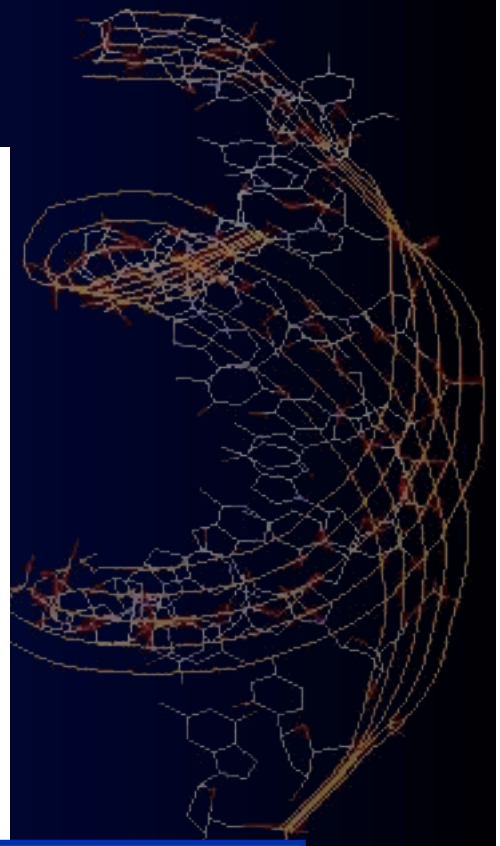
La linea :

```
printf("Io so programmare in C!");
```

è una chiamata di funzione.
In una chiamata di funzione, il nome della funzione è immediatamente seguito senza spazi da una parentesi aperta. Tra parentesi vi sono i dati da passare alla funzione (argomenti)

La funzione `printf`

```c
/* STAMPA1: stampa un numero */


#include <stdio.h>

main()
{
    printf("\n  Stampa un numero intero:\n\n");     /*stampa stringa <1>*/

    printf("%d",1234);                              /*stampa numero <2>*/
}

/* Note:
    <1> Se la stringa passata alla printf non contiene il carattere di
        percento %, viene stampata tale e quale.
    <2> Se la stringa contiene il carattere di percento %, questo indica
        l'inizio di una specifica di stampa. Ad esempio, %d indica di
        stampare un numero intero, che viene passato come secondo parametro.
        Dopo la stampa del numero, il cursore non torna a capo.
*/
```



Specifica di stampa

Funzione printf

`printf` e' una funzione della libreria stdio che permette ad un programma di passare informazioni allo standard output, il cui dispositivo associato e' tipicamente il terminale grafico. Il formato con cui sono presentati i numeri e le stringhe di caratteri e' definito da un insieme di regole e simboli di formattazione.

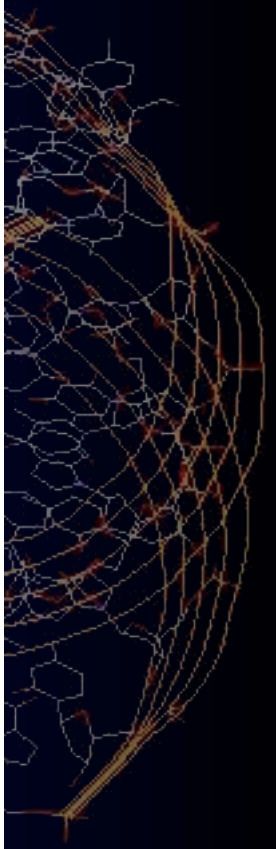
```
printf(formato, arg1, arg2….);
```

`formato` : stringa di caratteri che contiene le informazioni per la formattazione, i descrittori del formato

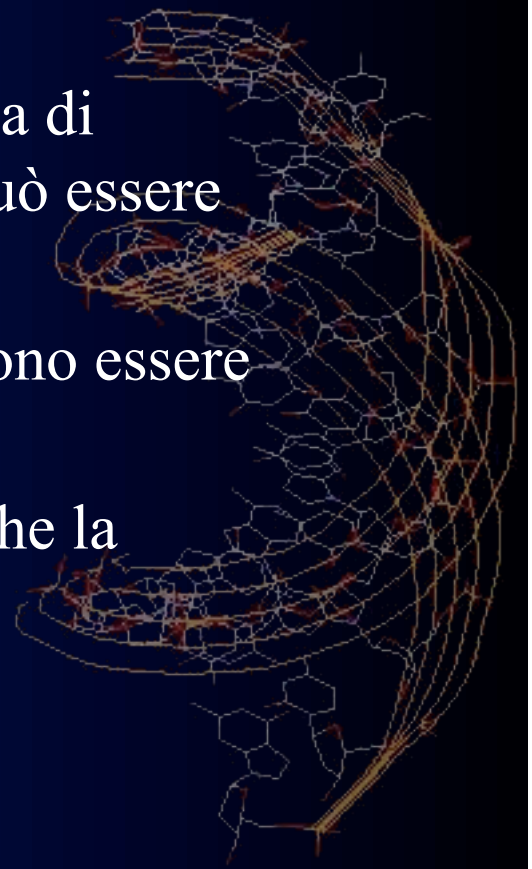
`arg1,arg2..`: nomi di variabili di cui si vuole stampare il contenuto

Esempio

```c
/* ALLINEA: stampe formattate di numeri con printf */


#include <stdio.h>

main()
{
    int x = 127;    /*numero da stampare*/

    printf("\n  Allineamenti di numeri con printf:\n\n");

    printf("Stampa normalmente:                  *%d*\n\n",x);     /*<1>*/

    printf("Allinea a destra in campo largo 8:   *%8d*\n",x);      /*<2a>*/
    printf("Allinea a sinistra in campo largo 8: *%-8d*\n\n",x);   /*<2b>*/

    printf("Stampa in campo troppo stretto (2):  *%2d*\n",x);      /*<3>*/
    printf("Stampa in campo largo 6, con zeri:   *%06d*\n",x);     /*<4a>*/
    printf("Stampa un minimo di 6 cifre:         *%.6d*\n\n",x);   /*<4b>*/

    printf("Stampa un numero negativo:           *%d*\n\n",-x);    /*<5>*/

    printf("Segno su numero positivo:            *%+d*\n",x);      /*<6a>*/
    printf("Segno su numero negativo:            *%+d*\n\n",-x);   /*<6b>*/

    printf("Spazio su numero positivo:           *% d*\n",x);      /*<7a>*/
    printf("Spazio su numero negativo:           *% d*\n",-x);     /*<7b>*/
}
```



Variabili e tipi

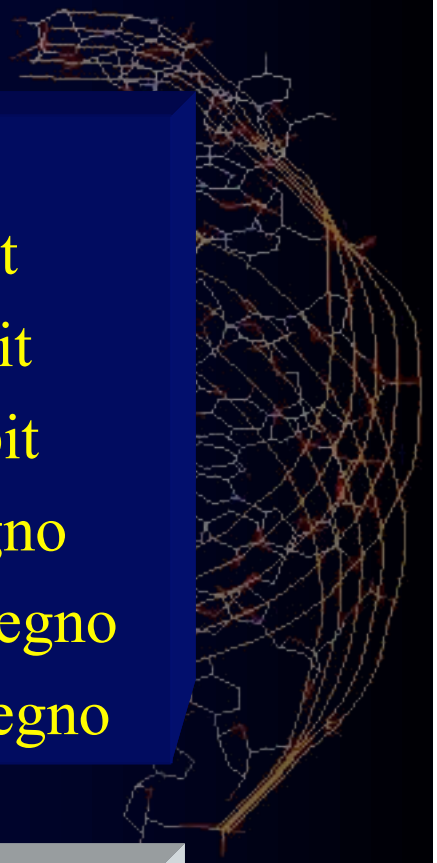


Ogni variabile corrisponde a una determinata cella di memoria utilizzata per contenere un valore che può essere modificato dal programma.

Tutte le variabili C prima di essere utilizzate devono essere dichiarate.

La dichiarzione serve per stabilire il tipo di dati che la variabile può contenere

Tipi semplici predefiniti in C



Tipi interi:

`int` = intero con segno, 32 bit

`short` = intero con segno, 16 bit

`long` = intero con segno, 64 bit

`unsigned int` = come int ma senza segno

`unsigned short` = come short ma senza segno

`unsigned long` = come long ma senza segno

I tipi character sono in realtà interi di 8 bit:

`char` = intero di 1 byte

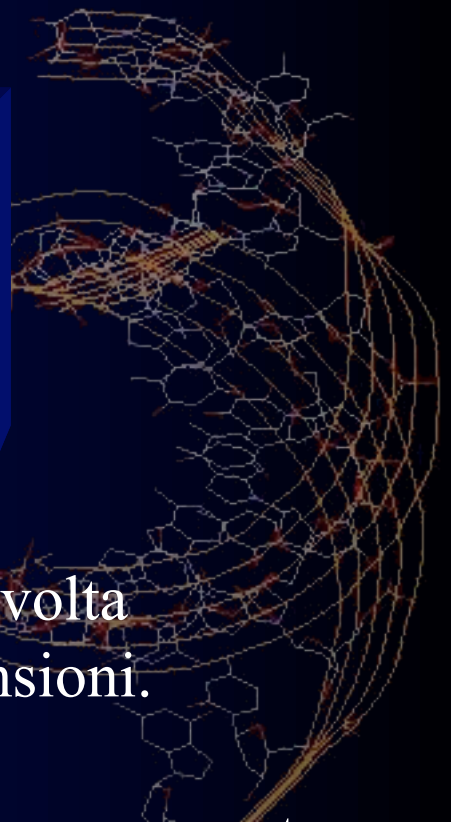
Tipi semplici predefiniti in C

Tipi float:

`float` = virgola mobile, 32 bit

`double` = virgola mobile, 64 bit

`long double` = virgola mobile, 128 bit



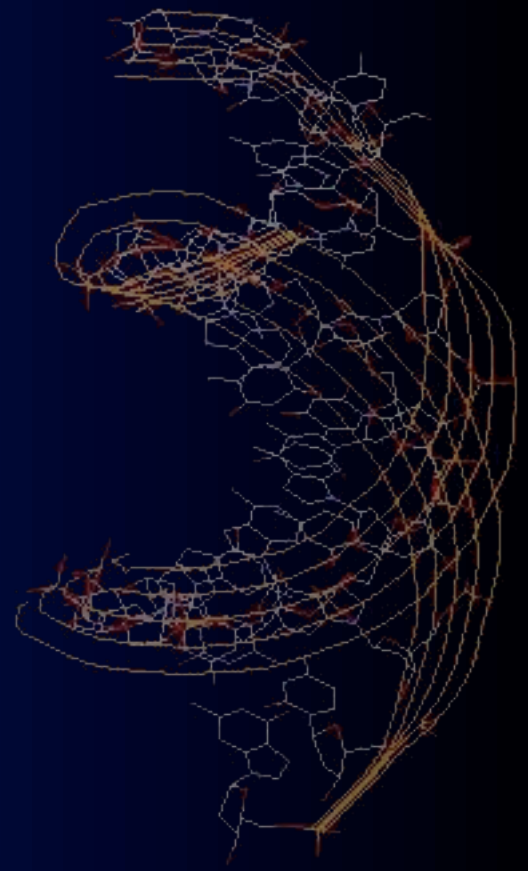
Il tipo più usato è il double, float è conveniente talvolta per risparmiare memoria in vettori di grandi dimensioni.

In aggiunta a questi tipi c'è il tipo speciale *void* che rappresenta l'assenza di valore (valore vuoto).

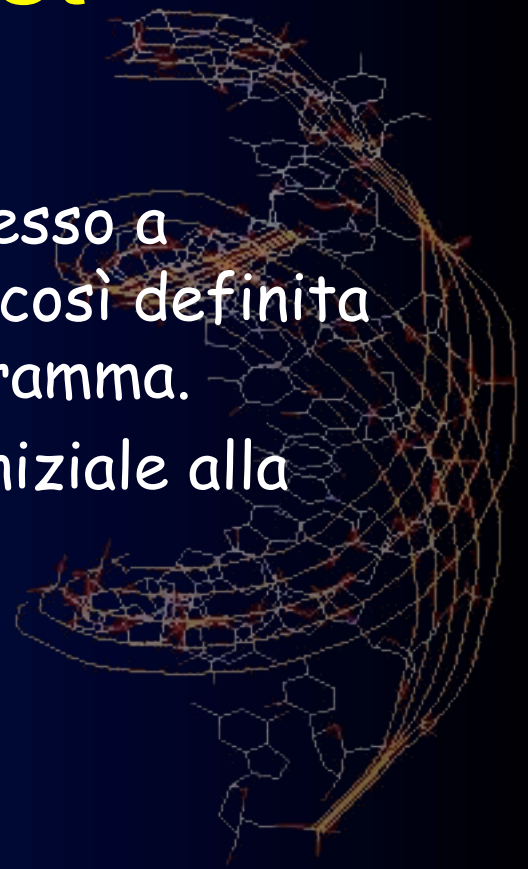
Vedremo meglio il suo utilizzo in seguito.

Esempio

```c
/* STAMPA2: stampa il valore di una variabile intera */


#include <stdio.h>

main()
{
    int anno;       /*dichiara che puï contenere un intero <1>*/

    printf("\n  Stampa il valore di una variabile numerica:\n\n");

    anno = 1993;                                    /*assegna valore*/
    printf("Mercato comune: %d\n",anno);            /*e lo stampa <2>*/
}

/* Note:
    <1> Dopo questa dichiarazione, la variabile anno puï essere usata
        liberamente. Il compilatore controlla comunque che non si cerchi
        di mettervi un dato del tipo sbagliato (cioä non un int).
    <2> La printf stampa come al solito la stringa che le viene passata,
        ma la specifica di stampa %d viene sostituita con il valore della
        variabile passata come secondo argomento (anno).
*/
```



Il qualificatore const



Il qualificatore ***const*** può essere premesso a qualsiasi tipo, indicando che la variabile così definita non verrà modificata nel corso del programma.

In tal caso bisogna assegnare il valore iniziale alla variabile in fase di definizione:

```
const int max_iterazioni = 100;
const char no = 'n';
const float pigreco = 3.141592;
```

Operatori aritmetici, relazionali e logici

Operatori aritmetici:

+ - * / %

Operatore modulo:

x%y è il resto della divisione di x per y e quindi vale zero se x è multiplo di y

Operatori relazionali:

> >= < <= == !=

Operatori di uguaglianza e disuguaglianza:

x = = y è vera se x è uguale a y

x != y è vera se x è diverso da y

Operatori logici:

&& ||

AND logico

OR logico

Operatori di incremento e decremento

Spesso in un programma occorre incrementare o decrementare di uno una variabile. Il C dispone di particolari operatori dedicati a ciò, che sono:

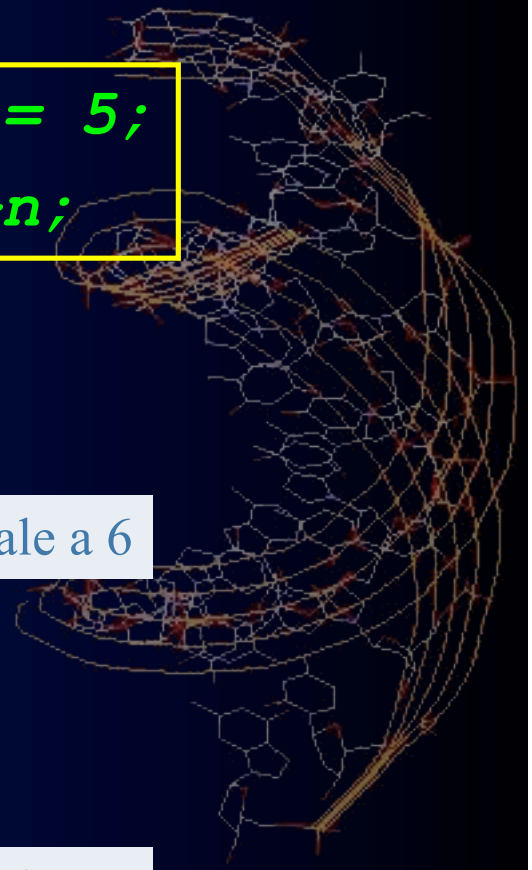
Se usati rispettivamente prima o dopo di una variabile ne determinano il preincremento (variabile prima incrementata e poi utilizzata) o il postincremento (variabile prima utilizzata e poi incrementata).

Esempi

```
n = 5;
n++;
```

→ Da ora in poi n è uguale a 6 ←

```
n = 5;
++n;
```



```
n = 5;
k = n++;
```

→ Da ora in poi k è uguale a 5 ed n è uguale a 6

```
n = 5;
k = ++n;
```

→ Da ora in poi k è uguale a 6 ed n è uguale a 6

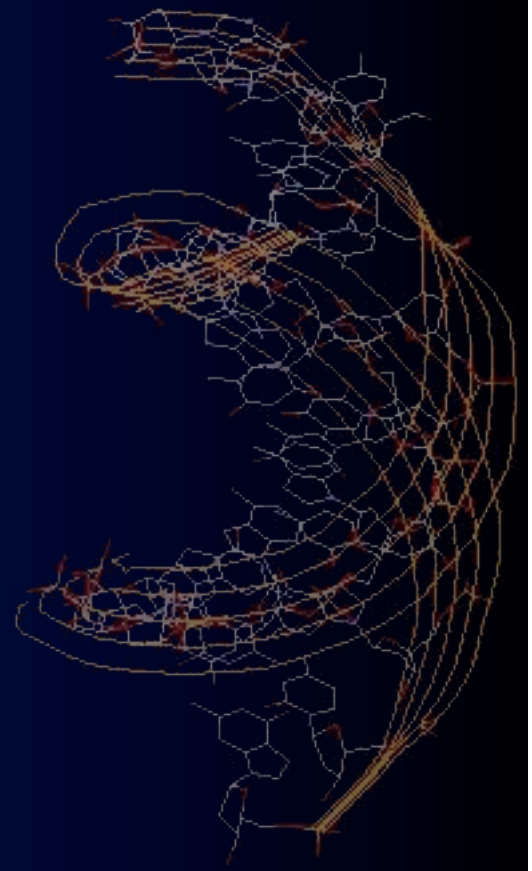
Operatori di assegnamento

Un'altra particolare classe di operatori del C permette di scrivere in forma compatta le operazioni in cui una variabile viene modificata a partire da sé stessa. Ad esempio:

`x += 2`	è equivalente a	`x = x + 2`
`x -=3`	è equivalente a	`x = x - 3`
`y *= x`	è equivalente a	`y = y*x`
`y *= x + 1`	è equivalente a	`y = y*(x + 1)`
`z %= x`	è equivalente a	`z = z%x`

La funzione `scanf`

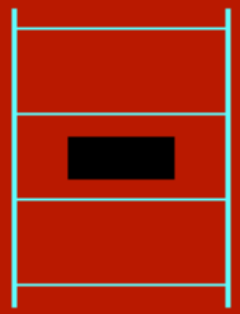
```c
/* DOPPIO: raddoppia il numero introdotto da tastiera */


#include <stdio.h>

main()
{
    int a;      /*numero da raddoppiare*/

    printf("\nNumero da raddoppiare? ");          /*prompt <1>*/
    scanf("%d",&a);                               /*input numero <2>*/

    printf("Il doppio di %d è %d\n",a,2*a);       /*calcola e stampa <3>*/
}

/*  Note:
    <1> Prima di un input, è sempre ben dire all'utente cosa ci si aspetta
        che venga introdotto da tastiera (notare che non va a capo).
    <2> Nella scanf non si indica la variabile a, ma il suo indirizzo &a;
        in questo modo la scanf sa dove si trova la variabile a nella
        memoria del computer, e può quindi modificarne il contenuto.
    <3> Per ogni specifica di stampa %d, dev'essere poi passato un valore;
        in questo caso i due valori passati sono a e 2*a.
*/
```



Indirizzo di una variabile

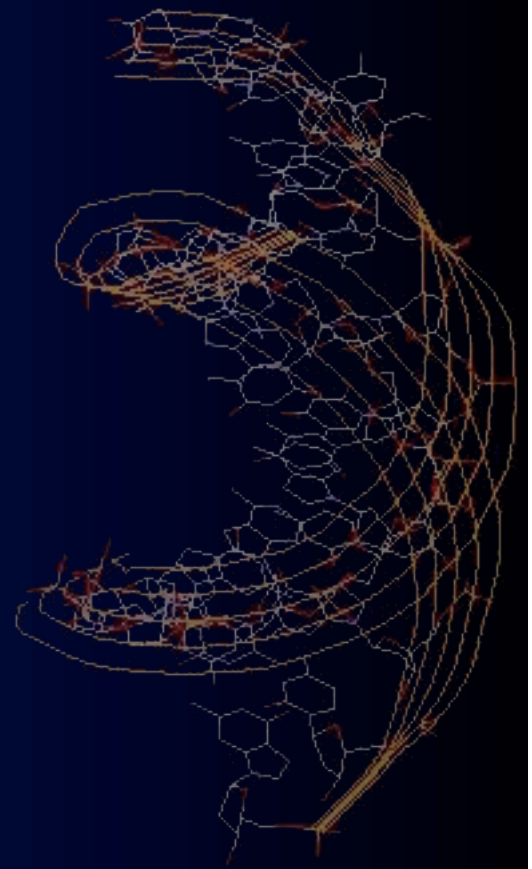
L'espressione `&a` indica l'indirizzo di `a` nella memoria

La memoria è vista come una serie di caselle numerate:



riservata alla variabile a

Quando si dichiara una variabile (es: int a;), il compilatore riserva una casella per la variabile.



Funzione printf

La stringa di controllo e' costituita da caratteri ordinari e da direttive per il controllo del formato.

Il descrittore del formato ha la seguente forma:

```
%[-][ampiezza][.precisione]tipo
```

`-` → allineamento del campo a sinistra (default a destra)

`ampiezza` → il numero totale di campi di carattere che il dato occupera';

Funzione printf

`precisione` →

- il numero max di caratteri da stampare per una stringa;
- il minimo numero di cifre da stampare per un intero
- il numero di cifre decimali per un numero floating point

`tipo` →

`d` per dati di tipo `int`;

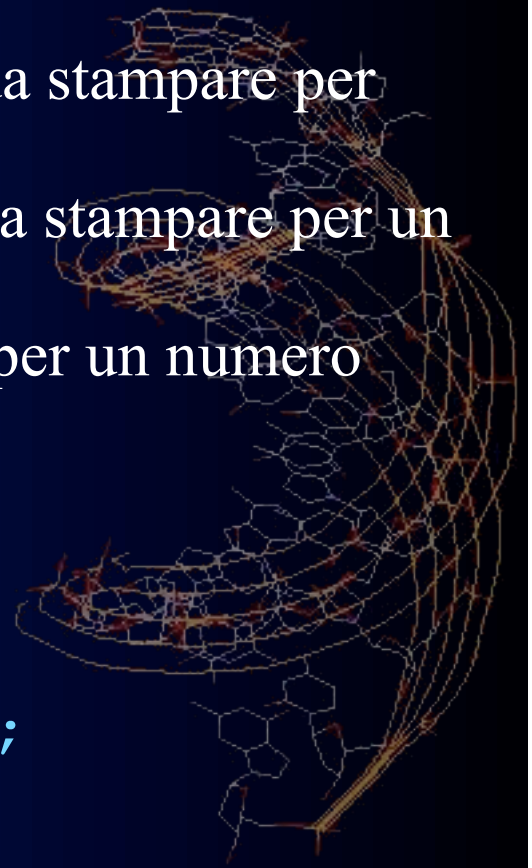
`f` per dati di tipo `float`;

`lf` per dati di tipo `double`;

`c` per dati di tipo `char`;

`s` per le stringhe

`e,E,g` per dati di tipo `double` con notazione scientifica;



Compilazione & linking con il compilatore GNU

```
$>gcc [-lnomelib] [-o file.exe] file.c
```

Cerca le librerie di nome:

lib*nomelib*.a

in una sequenza di directories di default ed effettua il linking.

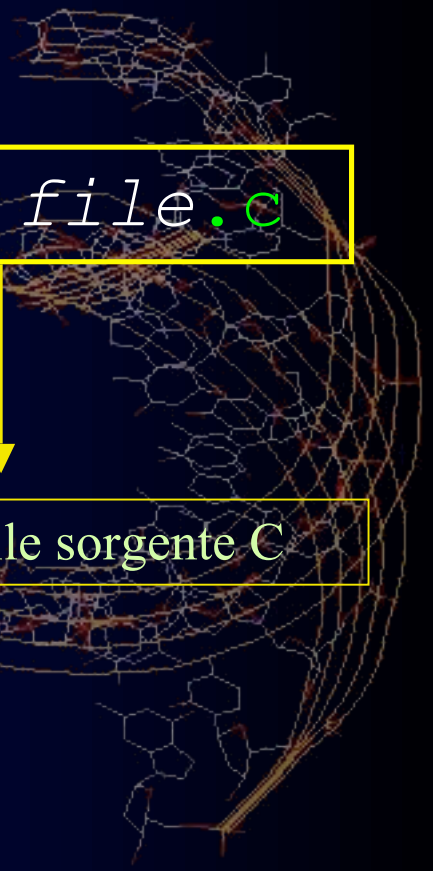
Esempio: -l*m* effettua il link con la libreria di funzioni matematiche

/usr/lib/lib*m*.a

Nome del file sorgente C

Nome del compilatore: include anche il linker

Genera in output un eseguibile chiamato *file*.exe invece del default, che è a.out



Un programma utile

```
/* Euroconvertitore */
#include<stdio.h>
main()
{
float cifra_in_lire, cifra_in_euro;
const float conv_fact = 1936.27;
printf("Inserisci la cifra in lire \n");
scanf("%f",&cifra_in_lire);
cifra_in_euro = cifra_in_lire/conv_fact;
printf("Il controvalore euro è %f \n",
cifra_in_euro);
}
```

Dichiarazione delle variabili.

Legge dallo standard Input un tipo float e lo mette nella variabile cifra_in_lire

Questo valore non può essere cambiato

Scrive sullo standard Output un messaggio contenente un tipo float che è il contenuto della variabile cifra_in_euro

Variabili e tipi

Come ampiamente discusso la rappresentazione interna di un numero intero con segno è diversa da quella di un intero senza segno.

Ci sono poi almeno tre possibili rappresentazioni dei numeri in virgola mobile secondo lo standard IEEE754: singola doppia e quadrupla precisione rispettivamente con 32, 64 e 128 bit.

I caratteri vengono rappresentati tramite il codice ASCII e necessitano di un byte (8 bit).

Il tipo di una variabile deve essere sempre dichiarato

Sommare patate con cipolle…

Nelle operazioni con tipi diversi, di norma un tipo inferiore viene "promosso" al tipo superiore secondo la scala:

`char,short=>int=>long=>float=>double=>long double`

Esempio:

Se ***`i`*** è un ***`int`*** e ***`f`*** è un ***`float`***, ***`i`*** viene promosso a ***`float`*** e il risultato dell'operazione ***`i+f`*** è a sua volta un ***`float`***

Bisogna ovviamente fare molta attenzione alle conversioni in cui si perde informazione, dove può avvenire un troncamento, come nel caso ***`i=x`*** con ***`i int`*** e ***`x float`***.

Input e Output di Caratteri

La libreria standard per l' I/O fornisce diverse funzioni per leggere e scrivere un carattere alla volta.

```
int c;

c = getchar();
```

la funzione getchar legge il successivo carattere disponibile nello standard input (es. tastiera)

```
putchar(c);
```

stampa nello standard output (es., terminale) il contenuto della variabile intera come carattere.

text stream: una sequenza di caratteri divisi in linee; ogni linea consiste di zero o piu' caratteri seguiti dal carattere di newline (\n).