

Alcune regole di base per scrivere un programma in linguaggio C

- Un programma in linguaggio C (listato) viene scritto in formato testo ed archiviato in un file: *.c
- Per scrivere un listato C si utilizza un qualsiasi programma di solo testo, ad esempio: Notepad; Edit; UltraEdit; Editor integrato nel pacchetto Dev-C++
- Il linguaggio C è “case-sensitive” nel senso che distingue tra lettere maiuscole e lettere minuscole. Ad esempio la variabile “area” è diversa dalla variabile “AREA” ed è ancora diversa dalla variabile “Area”.
- Fra una riga ed un'altra vi possono essere anche più righe vuote
- All'interno di una stessa riga vi possono essere più spazi vuoti o tabulazioni
- Ogni stringa di caratteri racchiusa fra /*.....*/ viene considerato commento, pertanto non ha nessuna influenza sul programma.

Struttura di un programma in C

/* Titolo */

Il titolo non è necessario,
se c'è deve essere fra */* .. */*

Direttive iniziali

①

- Dichiarazione di inclusione di (file header)
- Dichiarazioni di macro, costanti e/o variabili globali

main ()

②

Dichiarazione del nome della funzione principale che deve necessariamente essere *main*

{

Istruzioni

③

Corpo del programma sempre racchiuso fra {...}

}

① Dichiarazione di funzioni da richiamare

- In questa parte iniziale del programma vengono indicati i nomi dei file (file header) *.h che contengono l'elenco di eventuali funzioni che si vogliono richiamare ed utilizzare nel programma

`#include <nome-file-header-standard>`

`#include "nome-file-header-autocostruito"`

Se si utilizzano < > allora i file contenenti le funzioni vengono ricercate nelle directory standard del compilatore.

Se si utilizzano " " allora i file contenenti le funzioni vengono ricercate nella directory corrente.

Esempio:

`#include <stdio.h>`

`#include <math.h>`

- Dichiarazione di Macro, Costanti e/o Variabili globali.
Questa opportunità si sfrutta quando l'intero software è composto da più funzioni e le variabili e/o costanti vengono utilizzate in varie funzioni. Inizialmente ci limiteremo alla sola funzione main() pertanto non effettueremo dichiarazioni globali.

②

Dichiarazione del nome della funzione principale

- La funzione principale di ogni tipo di programma è la funzione “main()”. Ogni programma contiene quindi una funzione main(), all'interno della funzione main() possono essere richiamate altre funzioni standard o autoconstruite.

Tutte le istruzioni ed operazioni previste dalla funzione main() devono essere racchiuse all'interno di una coppia di parentesi { }

Esempio di programma completo che non fa nulla:

```
main ( )  
{  
}
```

③

Corpo del programma

- Il corpo del programma è fondamentalmente costituito da due parti:
 1. Dichiarazioni di variabili e/o costanti locali, cioè da utilizzare all'interno della funzione main()
 2. Successione di istruzioni.

Precisazioni:

- Ogni Dichiarazione e/o Istruzione deve concludersi con un carattere “;”
- L'intero corpo del programma è contenuto all'interno delle parentesi { }
- Vi possono essere ulteriori coppie di { } che racchiudono blocchi di Istruzioni particolari.

Esempio:

```
main ( )  
{  
    int a;  
    int b=100;  
  
    a = b;  
    b = a*2;  
}
```

Funzioni

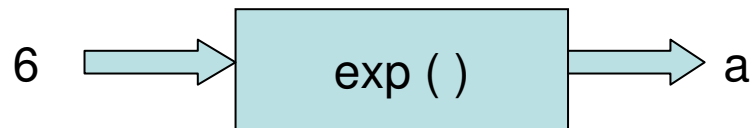
Molto spesso nella programmazione si utilizzano delle funzioni già pronte per eseguire operazioni ricorrenti.

Una funzione è caratterizzata da un nome e normalmente riceve uno o più dati (argomenti in ingresso) e ne restituisce altri.

Esempio:

```
#include <math.h>
main ( )
{
  float a;

  a = exp(6);
}
```

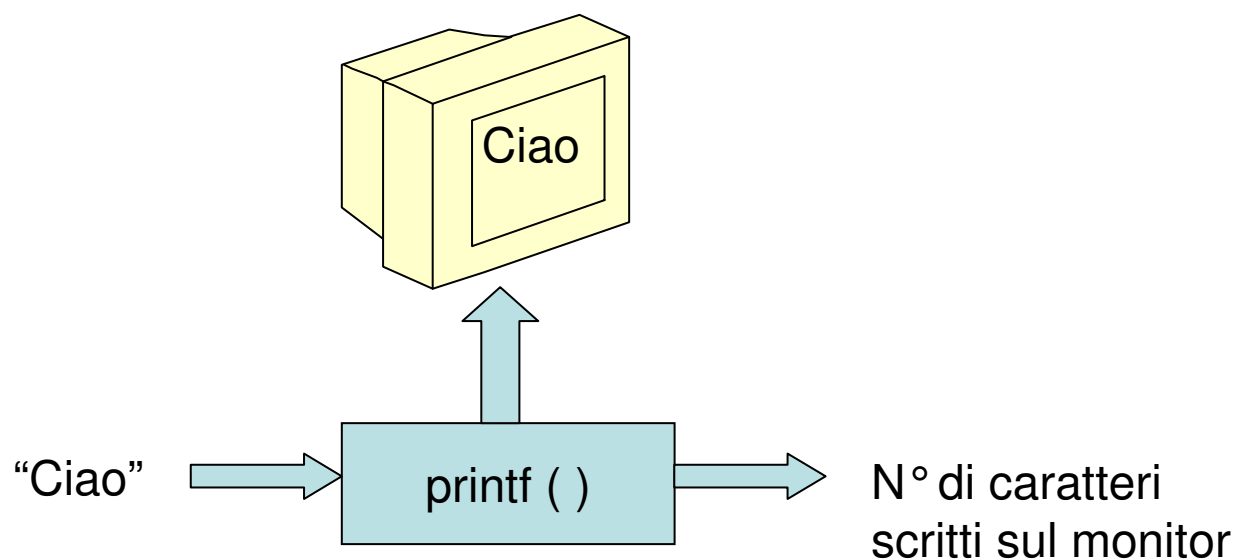


Funzione printf()

- La funzione printf () è probabilmente la funzione più richiamata in assoluto nei programmi scritti in C.
- La funzione printf () può essere utilizzata a condizione di dichiarare il file header “stdio.h” nella parte iniziale del listato.
- La funzione produce una scrittura sul monitor e spesso non si utilizza il valore numerico che restituisce alla funzione chiamante:

Esempio:

```
#include <stdio.h>
main ( )
{
    printf("Ciao");
}
```



Sequenze di escape con `printf()`

- La sequenza di caratteri racchiusa fra apici viene scritta sul monitor.
- Vi sono alcune sequenze di caratteri particolari che apparentemente non vengono stampati sul monito ma che producono comunque il loro effetto. Consideriamo in prima analisi le “sequenze di escape”:
 - `\n` produce un a capo riga
 - `\t` produce l’inserimento di spazi vuoti (tabulazione)
 - `\"` stampa il carattere doppio apice “
 - `\\` stampa una bara contraria \
 - `\a` produce un segnale acustico

Esempio:

```
#include <stdio.h>
main ( )
{
    printf("Ciao \n grande programmatore \n");
    printf("Ciao \t grande programmatore \a");
}
```

Esercizio N°1 Far scrivere sul monitor la scritta:
Tre casettine dai tetti aguzzi

Esercizio N°2 Far scrivere sul monitor la scritta:
Tre
casettine
dai
tetti
Aguzzi

Esercizio N°3 Far scrivere sul monitor la scritta:
Tre casettine dai tetti aguzzi

Creare un file Ese_123.txt con all'interno il prodotto dei tre programmi

Dichiarazione di costanti da utilizzare nel programma

Fra le direttive iniziali vengono dichiarare le costanti da utilizzare nel programma. Le dichiarazioni hanno la seguente forma:

`#define "identificatore" "valore"`

Es: `#define PIGRECO 3.14`
`#define IVA 0.20`
`#define PROVINCIA "Mantova"`

Gli identificatori sono scelti dal programmatore e possono essere costituiti da uno o più carattere. Il primo carattere deve essere una lettera o un carattere di sottolineatura e i caratteri successivi possono essere delle lettere, cifre o caratteri di sottolineatura.

Per consuetudine gli identificatori delle costanti si scrivono in maiuscolo.

i valori di tipo carattere o stringhe di caratteri vanno racchiuse tra doppi apici "..."

il separatore fra interi e decimali viene indicato con il carattere "."

Dichiarazione Identificatori e Tipo da utilizzare nel programma

Il corpo del programma deve prevedere prima di tutto il blocco dichiarativo relativo ai nomi e tipo da attribuire alle variabili da utilizzare nel programma.

Le dichiarazioni hanno la seguente forma:

tipo “*identificatore*”;

Es: **int pippo;**
 float area;

Se si prevedono più variabili dello stesso tipo allora è possibile effettuare una dichiarazione cumulativa:

tipo “*identificatore1*”, “*identificatore2*”, “*identificatore3*”,;

Es: **int pippo, base, altezza;**

Gli identificatori sono scelti dal programmatore e possono essere costituiti da uno o più carattere. Il primo carattere deve essere una lettera o un carattere di sottolineatura e i caratteri successivi possono essere delle lettere, cifre o caratteri di sottolineatura.

Alcuni Tipi di dati definiti dal linguaggio C

	Tipo	Intervallo	Bit utilizzati
caratteri	<code>char</code>	-128 a 127	8 bit
interi	<code>int</code>	-2.147.483.648 a 2.147.483.647	32 bit
reali	<code>float</code>	Sei cifre decimali di precisione	32 bit
	<code>double</code>	Dieci cifre decimali di precisione	64 bit

Nota: Su alcuni sistemi operativi il tipo "int" utilizza solo 16bit

Modificatori di Tipo:

`unsigned` → utilizza tutti i bit per rappresentare solo numeri positivi

`long` → se possibile aumenta il numero di bit utilizzati (dipende dal sistema)

`short` → diminuisce il numero di bit utilizzati

Es: `short int base;` → utilizza 16 bit
`long double prezzo;` → utilizza 80 bit dieci cifre di precisione
`unsigned int costo;` → utilizza 32 bit ma l'intervallo è:
da 0 a 4294967295

Caso particolare:

Variabili costituite da stringhe di caratteri di tipo char

Le variabili che sono destinate a contenere un solo carattere sono dichiarate analogamente a quanto già visto per le variabili numeriche:

tipo “*identificatore*”;

Es: `char risposta;`

→ la variabile “risposta” potrà assumere come valore solo un carattere

Le variabili che sono destinate a contenere più carattere sono invece dichiarate come segue:

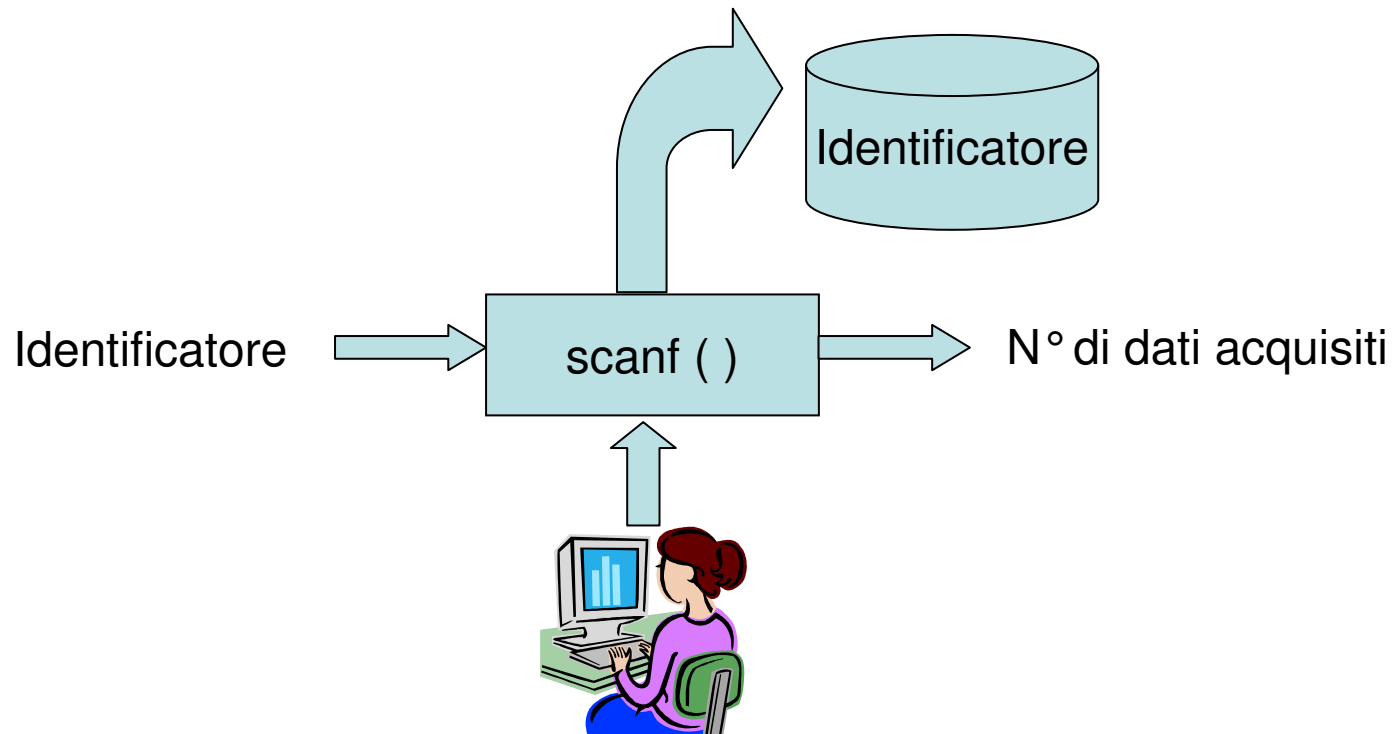
tipo “*identificatore [numero massimo di caratteri]*”;

Es: `char nome[10];`

→ la variabile “nome” potrà assumere come valore una stringa di caratteri di massimo 10 caratteri

Funzione `scanf()`

- La funzione `scanf ()` viene richiamata per acquisire dei dati da tastiera.
- La funzione `scanf ()` può essere utilizzata a condizione di dichiarare il file header “stdio.h” nella parte iniziale del listato.
- La funzione acquisisce un dato da tastiera e lo assegna alla variabile indicata in ingresso. **Normalmente non si utilizza il valore numerico che restituisce alla funzione chiamante (N° di dati acquisiti).**



Sintassi della funzione `scanf()`

- Alla funzione `scanf()` vengono passati almeno due parametri separati da virgole:

- il primo parametro rappresenta una stringa racchiusa tra apici costituita da specificatori di formato delle variabili corrispondenti che si vogliono acquisire.

Gli specificatori di formati sono delle sequenze di due caratteri e principalmente vengono usati i seguenti:

`%c` → Leggere un singolo carattere

`%d` → Legge un numero intero con segno

`%f` → Legge un numero reale

`%s` → Legge una stringa di caratteri

- i successivi parametri contengono i corrispondenti identificatori delle variabili che si vogliono acquisire dall'esterno.

Gli identificatori, che non siano stringhe di caratteri, devono essere preceduti dal simbolo “&”.

`scanf (“%...%.....”, &identificatore1, &identificatore2,);`

Nota: Molto spesso si utilizza `scanf()` per l'acquisizione di un solo valore per volta

Esempio di utilizzo di `scanf()`

Supponendo di voler acquisire dall'esterno il valore di una variabile "base" e di una variabile "altezza".

Si procede per semplicità con due distinte chiamate della funzione `scanf()` precedute da una opportuna richiesta di dati per l'utente.

```
/* Prova acquisizione dati */
```

```
#include <stdio.h>
```

```
main ( )
```

```
{
```

```
int base, altezza;
```

```
printf("Inserire base: ");
```

```
scanf("%d", &base);
```

```
printf("Inserire altezza: ");
```

```
scanf("%d", &altezza);
```

```
}
```

Caso particolare: scanf() per l'acquisizione di stringhe

- Per acquisire delle stringhe di caratteri, si utilizza lo specificatore di formato %s ed l'identificatore della variabile che conterrà la stringa non deve essere preceduta dal simbolo "&"

scanf ("%s", identificatore);

Esempio: Supponiamo di voler acquisire all'interno della variabile "nome" la stringa di caratteri del nome dell'utente:

```
/* Richiesta nome */  
  
#include <stdio.h>  
main ( )  
{  
  char nome[20];  
    printf("Inserisci il tuo nome: ");  
    scanf("%s", nome);  
}
```

Specificatore di formato in `printf()`

- Approfondiamo le potenzialità della funzione `printf()`
- La funzione `printf ()` accetta in ingrosso uno o più argomenti, Il primo argomento è rappresentato dalla stringhe di caratteri racchiusa fra due apici, gli altri argomenti sono etichette di variabili e/o costanti.
- Gli argomenti sono tutti separati da una virgola.
- All'interno della stringa vi sono delle sequenze di caratteri (specificatori di formato) che definiscono il modo e la posizione in cui dovranno essere visualizzati gli argomenti corrispondenti successivi.
- Gli specificatori di formato più utilizzati sono:
 - `%c` Carattere
 - `%d` Intero con segno
 - `%f` Numeri reali
 - `%e` Notazione scientifica
 - `%s` Stringa di caratteri

`printf("Stringa di caratteri", identificatore1, identificatore2,);`

Nel punto in cui viene inserito nella stringa uno specificatore di formato, comparirà sullo schermo il valore della corrispondente variabile indicate dagli identificatori successivi

Esempio di utilizzo di `printf()` con la stampa di variabili

Riprendiamo l'esempio precedente ed inseriamo una stampa dei valori delle variabili Introdotte dall'utente.

```
/* Prova acquisizione dati N°6 */
```

```
#include <stdio.h>
```

```
main ( )
```

```
{
```

```
int base, altezza;
```

```
printf("Inserire base: ");
```

```
scanf("%d",&base);
```

```
printf("Inserire altezza: ");
```

```
scanf("%d",&altezza);
```

```
printf("il valore della base inserita: %d \n", base);
```

```
printf("il valore della altezza inserita: %d \n", altezza);
```

```
}
```

Esempio di utilizzo di `printf()` e `scanf()`

Si vuole acquisire il nome, cognome ed età dell'utente per poi stamparli sul monitor.

```
/* Acquisizione dati Utente N°7 */
```

```
#include <stdio.h>
```

```
main ( )
```

```
{
```

```
int anni;
```

```
char nome[25], cognome[20];
```

```
printf("Inserire il nome: ");
```

```
scanf("%s", nome);
```

```
printf("Inserire il cognome: ");
```

```
scanf("%s", cognome);
```

```
printf("Inserire l'età: ");
```

```
scanf("%d", &anni);
```

```
printf("\n\n*****\n\n");
```

```
printf("\tNome:  \t%s\n",nome);
```

```
printf("\tCognome: \t%s\n",cognome);
```

```
printf("\tEtà:  \t%d\n",anni);
```

```
printf("\n\n*****\n\n");
```

```
}
```

Prova N°8

Si vuole acquisire da tastiera la Scuola, l'Indirizzo, la classe ed il numero di alunni della classe per poi stamparli sul monitor come qui di seguito indicato.

Nell'ipotesi che l'utente inserisco:

Scuola → Istituto Tecnico Commerciale
Indirizzo → Mercurio
Classe → 3B
N_Alunni → 21

Istituto Tecnico Commerciale ad indirizzo Mercurio

Classe: 3B

Numero Alunni: 21

Assegnazione di valori a variabili

- Una variabile può essere modificata assegnandogli un'opportuno valore:

Identificatore = valore;

- Il valore può essere o un numero oppure il risultato di un'opportuna operazione fra variabili e/o numeri.
- Gli operatori algebrici utilizzabili in C sono:

+ Somma

- Sottrazione

/ Divisione

* Moltiplicazione

% Calcolo del resto della divisione fra due interi

Esempio:

.....

int a,b,c;

float x,y,z;

a = 29;

/* la variabile a assumerà il valore 29 */

b = 8;

/* la variabile b assumerà il valore 8 */

c = a*b;

/* la variabile c assumerà il valore 232 */

x = a%b;

/* la variabile x assumerà il valore 5 */

z = x - a;

/* la variabile z assumerà il valore -24 */

.....

Esempio di utilizzo delle istruzioni di assegnazione

Si vuole scrivere un programma che dato il raggio di un cerchio ne stampi la lunghezza della circonferenza e l'area.

```
/*  
Cerchio.c --> Calcola la lunghezza della circonferenza e  
l'area di un cerchio avente raggio dato dall'utente  
*/  
  
#include <stdio.h>  
#define PIGRECO 3.14  
  

```