LABORATORIO DI FONDAMENTI DI INFORMATICA
28 SETTEMBRE 2020 - Incontro 1 di 5 – Introduzione (printf, scanf, if, for)

PROGRAMMA DI ESEMPIO

```
#include <stdio.h>
/* Questo programma chiede all'utente un numero intero e lo
ristampa a video */
int i;
int main() {
     printf("Numero? ");
     scanf("%d", &i);
     printf("E' stato inserito il numero intero %d.\n", i);
     return 0;
}
```

ESERCIZIO 1

Scrivere un programma che stampi a video il seguente messaggio su due righe:

```
Hello World!
Ciao Mondo!
```

All'inizio del programma inserire un commento che contenga il nome dell'autore e la data di creazione del codice.

Suggerimenti:

- `int main()` è la speciale funzione che indica dove inizia l'esecuzione del programma e deve restituire un numero intero. Occorre quindi ricordarsi di utilizzare `return` per restituire il numero `0` che per convenzione indica che non si sono verificati errori.
- La funzione per stampare sullo standard output (cioè a video) è `printf` e per utilizzarla occorre includere la libreria di standard input/output con la direttiva al preprocessore
  ```
  #include <stdio.h>
  ```
- per andare a capo nella stampa tramite `printf` si utilizza la "escape sequence" `\n`
- I commenti su una riga iniziano con `//`
  ```
  // Commento su una riga
  ```
- I commenti multiriga si aprono con `/*` e si chiudono con `*/`
  ```
  /* Commento Multiriga */
  ```

ESERCIZIO 2

Scrivere un programma che dichiari tre variabili intere

```
a, b, somma
```

e chieda all'utente di inserire i valori di `a` e `b` (indifferentemente in base 10 o base 8 o base 16).

Esempio in base 10 in cui l'utente utilizza le cifre 0-9:

```
a? 2
b? 8
```

Esempio in base 8 in cui l'utente antepone il prefisso zero `0` al numero e utilizza le cifre 0-7:

```
a? 02
b? 010
```

Esempio in base 16 in cui l'utente antepone il prefisso `0x` al numero e utilizza le cifre 0-9 e A-F:

```
a? 0x2
b? 0x8
```

Il programma deve assegnare alla variabile `somma` il risultato di `a + b` e stampare a video i valori delle 3 variabili (`a, b, somma`) in base 10, base 8 e base 16 con il seguente messaggio:

```
Base10: a + b = 2 + 8 = 10
Base8 : a + b = 2 + 10 = 12
```

```
Base16: a + b = 2 + 8 = a
```

Suggerimenti:

- la funzione per richiedere i valori delle variabili all'utente tramite standard input (cioè la tastiera) è `scanf`. Occorre ricordare di anteporre il simbolo `&` alla variabile che riceve il valore, ad esempio:
  ```
  scanf("%d", &a);
  ```
- i "segnaposto" da usare in `printf` e/o `scanf` per i numeri interi sono i seguenti:
  - `%d` per i numeri interi in base 10
  - `%o` per i numeri interi in base 8
  - `%x` per i numeri interi in base 16
  - `%i` indica a `scanf` che può accettare in input numeri interi sia in base 10 che 8 che 16 a seconda di quanto indicato dall'utente (che potrà usare il prefisso `0` per la base 8 e il prefisso `0x` per la base 16)

ESERCIZIO 3

Scrivere un programma che dichiari le seguenti variabili:

```
int x = 3;
float y = 3.0;
float r1, r2, r3;
```

ed esegua le seguenti operazioni:

```
r1 = x / 2;
r2 = x / 2.0;
r3 = y / 2;
```

Riflettere su quale sarà il valore inserito nelle variabili `r1, r2 e r3`.
Verificare se l'ipotesi fatta è corretta stampando a video le variabili `r1, r2, r3`.
Provare a stamparle anche limitando la parte decimale a 3 cifre.
Provare a stamparle anche in notazione scientifica.

```
r1=1.000000 r2=1.500000 r3=1.500000
3 cifre decimali: r1=1.000 r2=1.500 r3=1.500
Notazione scientifica: r1=1.000e+000 r2=1.500e+000 r3=1.500e+000
```

Suggerimenti:

- il segnaposto per i numeri decimali e `%f`
- è possibile stabilire il numero di colonne da utilizzare per stampare la parte intera e/o decimale di un numero indicandolo nei segnaposto:
  - `%3d` utilizza 3 colonne per il numero intero
  - `%.3f` numero automatico di colonne per la parte intera e 3 per la parte decimale
  - `%5.3f` 5 colonne per la parte intera e 3 per la parte decimale
- il segnaposto per la notazione scientifica è `%e`
- il segnaposto `%g` lascia decidere alla funzione `printf` se sia preferibile utilizzare la notazione normale o quella scientifica a seconda del valore da stampare

ESERCIZIO 4

Scrivere un programma in cui si dichiarino due variabili adatte a contenere dei caratteri, una per l'iniziale del nome e una per l'iniziale del cognome

```
inome, icognome
```

e si chiedano all'utente le lettere iniziali del suo nome e cognome per poi ristamparle a video

```
Iniziale del nome? p
Iniziale del cognome? v
Le tue iniziali sono: p.v.
```

Suggerimenti:

- In questo caso per l'input si possono usare indifferentemente le funzioni `getchar` o `scanf`

- Potrebbe accadere che il programma non attenda l'inserimento dell'iniziale del cognome e che venga stampata direttamente solo l'iniziale del nome:
  ```
  Iniziale del nome? p
  Iniziale del cognome? Le tue iniziali sono: p.
  ```
  Questo accade perché la seconda lettura di carattere registra il carattere di "a capo" che è stato introdotto premendo il tasto "invio".
  Ci sono 3 possibili soluzioni per evitare che questo accada:
  - eseguire una ulteriore lettura tramite `scanf` (o `getchar`) per "intercettare" il carattere di a capo e ignorarlo
    ```
    getchar();
    scanf("%c", &icognome);
    ```
  - svuotare il buffer della tastiera prima di leggere un nuovo carattere
    ```
    fflush(stdin);
    scanf("%c", &icognome);
    ```
  - aggiungere uno spazio prima del segnaposto `%c` per indicare a `scanf` di ignorare il carattere di a capo
    ```
    scanf(" %c", &icognome);
    ```

ESERCIZIO 5

La funzione `sizeof` permette di poter conoscere la quantità di memoria occupata per ciascun tipo di dato (ad esempio `int`). Tale quantità può variare in base al sistema o al compilatore.
`sizeof` utilizza come unità base di misura la quantità di memoria occupata dal tipo `char` `sizeof(char)` varrà quindi sempre 1 e se `sizeof(int)` vale 4 significa che il tipo `int` richiede 4 volte la memoria richiesta dal tipo `char`.
Per conoscere la quantità di memoria in bit effettivamente riservata per il tipo `char` occorre includere una libreria utilizzando la direttiva

```
#include <limits.h>
```

che mette a disposizione la costante intera

```
CHAR_BIT
```

Solitamente `CHAR_BIT` vale 8 (cioè 8 bit).
Scrivere un programma che stampi a video la quantità di memoria occupata nel proprio sistema per i seguenti tipi:

```
char
int
long int
float
double
long double
```

Il risultato sarà simile al seguente (ma potrebbe differire a seconda del proprio sistema):

```
CHAR_BIT = 8 bit
char: 1 * 8 bit = 8 bit
int: 4 * 8 bit = 32 bit
long int: 4 * 8 bit = 32 bit
float: 4 * 8 bit = 32 bit
double: 8 * 8 bit = 64 bit
long double: 16 * 8 bit = 128 bit
```

Suggerimenti:

- Il compilatore potrebbe segnalare un warning per indicare che il segnaposto `%d` non è adatto al tipo di intero ritornato da `sizeof` perché molto più grande di `int`.
  In questo caso è possibile utilizzare l'operatore di cast `(int)` per eseguire una conversione.
  ```
  printf("int: %d\n", (int) sizeof(int));
  ```

ESERCIZIO 6

Scrivere un programma che richieda all'utente un numero intero `num` e stampi a video il risultato della sua divisione intera per 2 e indichi se `num` sia pari o dispari.

```
Inserisci un numero intero: 5
Il resto della divisione intera 5 % 2 e' 1.
Hai inserito il numero 5 che e' un numero dispari.
```

Suggerimenti:

- un numero intero è pari se il resto della divisione per 2 è uguale a 0, è dispari se il resto è 1. Nel linguaggio C l'operatore che restituisce il resto di una divisione tra numeri interi è `%`.
- attenzione a non confondere l'operatore di assegnamento = e l'operatore di confronto ==
- per stampare con `printf` il simbolo `%` occorre inserirlo "raddoppiato"

```
printf(" %% ");
```

ESERCIZIO 7

Scrivere un programma che simuli il gioco "carta-forbice-sasso".

Chiedere a ciascuno dei due giocatori un carattere che indichi quale oggetto desideri giocare (`'c'` = carta, `'f'` = forbici, `'s'` = sasso) e indicare il risultato della partita ("pari" o "vince il giocatore 1 / 2").

Si ricorda che se entrambi i giocatori giocano lo stesso oggetto la partita è pari, che il sasso vince sulle forbici, che le forbici vincono sulla carta, e che la carta vince sul sasso.

NB: Occorre inserire una ulteriore lettura "`scanf`" o una istruzione "`fflush(stdin);`" per ripulire il buffer della tastiera tra le due domande di input; serve per "intercettare" il carattere di "invio" altrimenti il programma sembra non attendere l'inserimento del giocatore 2.

```
Oggetto giocatore 1? c
[Svuotamento del Buffer della tastiera...]
Oggetto giocatore 2? f
Le forbici tagliano la carta - vince il giocatore 2.
```

Suggerimenti:

- i caratteri in C vanno racchiusi tra apici singoli

```
if (giocatore1 == 'c' && ...
```

ESERCIZIO 8

Scrivere un programma che chieda un numero da 1 a 10 all'utente e ne stampi la tabellina.

```
Numero? 3
Tabellina del numero 3:
3 x 1 = 3; 3 x 2 = 6; 3 x 3 = 9; 3 x 4 = 12; 3 x 5 = 15;
3 x 6 = 18; 3 x 7 = 21; 3 x 8 = 24; 3 x 9 = 27; 3 x 10 = 30;
```

Suggerimenti:

- si consiglia di implementare un ciclo `for`

ESERCIZIO 9

Scrivere un programma che chieda all'utente un numero intero `n` > 0 e stampi a video la somma dei primi `n` numeri interi.

```
Numero? 4
Somma [1,4] = 1 + 2 + 3 + 4 = 10
```