Fondamenti di Informatica T-1
Parte 2

Laboratorio 01

Obiettivi di questa esercitazione

1. Input e output in linguaggio C
2. Espressioni
3. Programmazione strutturata 1 (if, switch, do…while)

Warning vs Errors (1)

- Il risultato del processo di compilazione è:
  - se il programma è **sintatticamente corretto** → un file oggetto, pronto per essere sottoposto alla fase di *linking*
  - altrimenti → la notifica di una serie di errori
- In entrambi i casi, il compilatore può comunque segnalare dei warning
  - *Potenziali* sorgenti di errori a run-time o di comportamenti non voluti

Warning vs Errors (2)

- Ci sono *warning* più o meno gravi
  - Alcuni non rappresentano un vero problema…
  - …spesso, invece, possono essere una vera fonte di errori a tempo di esecuzione
    - Nota che gli errori che si verificano runtime sono i più difficili da prevedere/gestire
  - Il livello di gravità di un *warning* può essere valutato solo dal programmatore, dipendentemente dal contesto in cui si verifica
- Quindi È FONDAMENTALE CONSIDERARE CON ATTENZIONE TUTTI I WARNING EVENTUALMENTE GENERATI DAL COMPILATORE

Warning vs Errors (3)

Un semplice programma:

```
int main(void)
{
  float IVAperc = 0.2;
  float prezzo = 11200;
  printf("IVA %f", IVAperc * prezzo);
  return (0);
}
```


Warning vs Errors (4)

- **`float IVAperc = 0.2;`**
  - *warning C4305: 'initializing' : truncation from 'double' to 'float'*
  - Non c'è in realtà alcun troncamento (0.2 non richiede doppia precisione) ma a default le costanti numeriche reali vengono tradotte in double
  - Conviene seguire la sintassi corretta:
    `float f = 0.2f;`
- Supponiamo ora di modificare la definizione in
  `int IVAperc = 0.2;`
  - *warning C4244: 'initializing' : truncation from 'double' to 'int', possible loss of data*
  - 0.2 viene in effetti troncato a 0
  - Nell'operazione **`IVAperc * prezzo`**, **`IVAperc`** viene promosso a float, ma ormai ***la perdita di informazione è avvenuta***
  - ***Il risultato è sempre e comunque 0 (GRAVE)!***

Input e output in C (1)

- Input con formato:

  scanf(“stringa formato”, lista variabili);
- Output con formato:

  printf(“stringa formato”, lista variabili);
  - Tramite la stringa di formato, si specifica “come”
  - Tramite i parametri successivi, si specifica “che cosa”

Esempio 1 input output

Esempio 1 – Echo di un numero intero

- Realizzare (cioè scrivere e compilare) un programma che legga da tastiera un numero intero e ne stampi il valore a video (echo)
- pseudo-algoritmo:
  - Leggo da input un numero intero
  - Salvo il numero letto in una variabile apposita
  - Stampo a video il valore della variabile

Esempio 1 - Soluzione

```
#include <stdio.h>
int main()
{
  int value;

  scanf(“%d”, &value);
  printf(“Valore letto:%d\n”, value);
  return 0;
}
```


Esercizio 1

Esercizio 1 – Echo di caratteri

- Realizzare un programma che legga da tastiera tre caratteri e ne stampi il valore a video (echo)

Esempio 2 espressioni condizionali

Esempio 2 – Stabilire il massimo tra due valori

- Realizzare un programma che legga da input due numeri reali, e ne stampi a video il valore massimo
- Al fine di determinare il massimo, si utilizzino le sole espressioni condizionali

Esempio 2 - Soluzione

```
#include <stdio.h>
int main()
{
  float num1, num2, max;

  scanf("%f %f", &num1, &num2);

  max = ((num1 > num2) ? num1 : num2);
  printf("Max: %f\n", max);
  return 0;
}
```

Esercizio 2

Esercizio 2 – Elaborazione di numeri reali

- Realizzare un programma che legga da input un numero reale, e stampi a video:
  1. il suo valore assoluto
  2. il valore assoluto della sua parte intera

Esercizio 3

Esercizio 3 – Stampa di caratteri in ordine alfabetico

- Realizzare un programma che legga da input tre caratteri e li stampi in ordine alfabetico.

A tal scopo, si rammenti la rappresentazione dei caratteri in linguaggio C…

- Si utilizzino solo le espressioni condizionali (e non l'istruzione if… ad esempio)

Esempio 3 cicli

Esempio 3 – echo di caratteri

- Realizzare un programma che legga ciclicamente da input un carattere e lo stampi a video
- Il programma deve terminare quando l'utente inserisce il carattere '0'

Esempio 3 - Soluzione

```
#include <stdio.h>

int main(void)
{
  char c;
  do
  {   scanf("%c", &c);
      printf("%c", c);
  }   while(c > 0);
  return 0;
}
```

Esercizio 4 (1)

Esercizio 4 - Calcolo di un logaritmo in base qualunque

- Realizzare un programma che chieda all'utente due numeri razionali, uno detto base B e l'altro detto valore X (a cui applicare il logaritmo)
- Se e solo se ***B e X sono entrambi positivi***, il programma deve fornire come risposta il logaritmo in base B di X...
- ...altrimenti deve stampare un messaggio di errore

Esercizio 4 - Analisi del problema

- Si rammenti che, data una funzione logaritmica in base k, un logaritmo in una base diversa B è così definito:

$$\log_B(x) = \frac{\log_k(x)}{\log_k(B)}$$

- Approccio bottom-up: posso sfruttare del codice già fatto?
- Si consideri l'uso delle funzioni log(X) (logaritmo naturale) e log10(X), disponibili in **math.h**

Esercizio 4 - Schema di soluzione

Schema di soluzione