LEZIONE 6 - LABORATORIO

29 SETTEMBRE 2023

9-11 TURNO A-B
11-13 TURNO C

ho cercato di riportare il lavoro fatto in laboratorio, aggiungendo il lavoro fatto insieme sul codice, in modo che possiate ripetere tutto a casa.

1

ESERCIZI DEL FINE SETTIMANA

- Troverete sul sito del corso (e sul mio sito personale unisa) un elenco di esercizi suggeriti.
- Fateli
- Se incontrate difficoltà, contattatemi.
- Non vi dimenticate che difficoltà tecniche per l'installazione di gcc sono affrontate nella guida caricata sul sito del corso (e sul sito personale unisa).

2

PER GLI ACCOUNT IN LAB

- il primo accesso deve essere fatto a Windows
- poi potrete usare cosa preferite
- Abituatevi a lavorare su queste macchine su entrambi gli ambienti, ma sempre usando: editor di testo + finestra del terminale (prompt del dos, terminale, cygwin, windows power shell)
- stiamo preparando l'ambiente esame

3

OGGI

- comandi linux (qualcosa)
- uno/due esercizi guidati di programmazione (Programming Projects)

spero che abbiate con voi
i nostri compagni di viaggio

compilatore C

4

EDITOR DI TESTO (non Word – è formattato)

+

finestra del terminale per eseguire gcc (come «sotto» Linux)

NON VISUAL STUDIO o altri IDE,
DOVETE AVERE TUTTI LA STESSA MODALITA' DI LAVORO
- ANCHE IN VISTA DELL'ESAME -

aiutatevi e/o contattatemi se avete problemi

5

ALCUNI COMANDI LINUX DAL TERMINALE

- *Per creare una cartella con nome X:* `mkdir X`
- *Per entrare nella cartella **X:*** `cd X`
- *Per restituire la directory corrente*: `pwd`
- *Per tornare nella home*: `cd`
- *Per risalire di un livello di directory*: `cd ..`
- *Per rimuovere un file*: `rm nome_file`
- *Per cambiare nome*: `mv nome_file nuovo_nome_file`
- *Per copiare un file*: `cp nome_file nuovo_nome_file`
- *Per visualizzare il contenuto della directory*: `ls`

Si può anche rinominare il file eseguibile:
gcc nome_file.c -o nome_file

- ***Per compilare***: gcc nome_file.c [per default, l'eseguibile è in a.out]
- ***Per eseguire***: ./a.out

In questo caso: ./nome_file

6

BREVE RIASSUNTO DI QUANTO FATTO IN AULA INSIEME

- Vi ho mostrato come accedere alle macchine virtuali del laboratorio tramite VMWare Client (seguire il pdf sul sito del corso)
  - Collegarsi a Ubuntu 20.04
  - A noi basta aprire un Terminale (se non è già sul desktop, lo cercate nelle applicazioni - in basso a destra compare l'icona) e un editor di testo, ad esempio gedit.

  Ovviamente potete usare cygwin+notepad, oppure WindowsPowerShell+Notepad, o qualsiasi altro editor di testo (senza completamente automatico).

- Nota: Se la tastiera è settata inglese e volete la tastiera italiana, dal terminale digitate questo comando

  setxkbmap it

7

PER FORZARE UN COMPILATORE

Non lo abbiamo visto in aula, ma potete forzare l'uso del compilatore c89 (oramai standard).

Questo può anche essere fatto in combinazione con l'opzione che forza la visualizzazione di tutti i warning se il vostro gcc non li mostrasse già. Basta digitare questo comando, invece del semplice gcc, per compilare il file esempio.c

gcc -Wall -std=c89 esempio.c

- (vedere Q&A del capitolo 2)

8

ABBIAMO GIOCATO UN PO' CON I COMANDI

- vi ho fatto vedere l'uso della **mkdir** e l'effetto sia sul terminare (con **ls**) sia in ambiente grafico con le icone.
- Attenzione ai nomi doppi: non usate sia per le directory, sia per i file, nomi con spazi o punti. Questi poi, usati da linea di comando, creano problemi.
- Abbiamo usato **cd** e **cd..** per spostarci nelle cartelle.
- E' importante l'uso di cd. Ricordate che quando compilate un file, gcc deve essere eseguito nella cartella dove è salvato il file (oppure il file deve trovarsi nella cartella dove avete gcc).
- Non abbiamo usato **cp** o **rm**, ma potete esercitarvi da soli.

9

LAYOUT DI UN PROGRAMMA IN C

Buone regole:

- *Dividere le istruzioni* su più linee
- *Spazio tra tokens* (alcuni sono detti "essenziali")
- *Indentazione*
- *Linee bianche* per dividere il programma in unità logiche.
- Commentare…

```
in t main(void);
/*** WRONG ***/
```

Ex. K,2.10: fatelo. Riguarda questo argomento

10

Esercizio 0

Scrivere un programma che stampi a video quanto segue (andando a capo alla fine). **Compilarlo**. **Eseguirlo**.

```
To C, or not to C: that is the question.
```

```
/* questo programma stampa a video una frase
   su una sola riga e va a capo */


int main(void)
{

  printf("To C, or not to C: that is the question.\n");
  return 0;
}
```

Ora compilate (gcc nomefile.c)

11

IL COMPILATORE DOVREBBE INDICARVI UN WARNING

- Non trova la definizione della printf
- Se vi dà errore (dipende dal gcc) sappiamo che dobbiamo sistemare il codice per farlo funzionare, altrimenti l'eseguibile a.out (o a.exe se siete con Cygwin) non viene creato
- Se dà warning, l'eseguibile viene creato e sembra tutto corretto.
- In generale i warning vanno tutti risolti. Aggiungiamo `#include <stdio.h>` che avevamo dimenticato

12

Esercizio 0

Scrivere un programma che stampi a video quanto segue (andando a capo alla fine). **Compilarlo**. **Eseguirlo**.

```
To C, or not to C: that is the question.
```

```
/* questo programma stampa a video una frase
   su una sola riga e va a capo */

#include <stdio.h>

int main(void)
{

  printf("To C, or not to C: that is the question.\n");
  return 0;
}
```

Ricompiliamo. Tutto ok. Ora eseguite (./a.out)

13

Esercizio 0

Scrivere un programma che stampi a video quanto segue (andando a capo alla fine). **Compilarlo**. **Eseguirlo**.

```
To C, or not to C: that is the question.
```

```
/* questo programma stampa a video una frase
   su una sola riga e va a capo */

#include <stdio.h>

int main(void)
{

  printf("To C, or not to C: that is the question.\n");
  return 0;
}
```

Ora eseguite (./a.out)

```
>To C, or not to C: that is the question.
>
```

14

ABBIAMO GIOCATO UN PO' SUL CODICE

- Inserendo e togliendo \n nella printf
- Provate a togliere return 0; (anche qui, non segnala errori/warning, ma per buona prassi, lasciatelo)
- Quando vogliamo togliere una istruzione, piuttosto che cancellarla, possiamo commentarla usando //,

```
//return 0;
```

15

weight.c

```
/* Calcola il volume di un parallelepipedo di dim
 8,10,12*/

#include <stdio.h>
int main(void)
{
  int height, length, width, volume;
  height = 8;
  length = 12;
  width = 10;
  volume = height * length * width;   //formula


  printf("Volume: %d\n", volume);


  return 0;
}
```

vedremo che possiamo eliminare qualcosa…

16

DOBBIAMO IMPARARE DAGLI ERRORI (E DAI WARNING)

- Abbiamo introdotto errori sintattici (togliere una dichiarazione, ad esempio) e abbiamo visto che il compilatore lo segnala.
- Abbiamo introdotto errori semantici, che ovviamente il compilatore non può segnalare.
- Ad esempio, non inizializzare una variabile (usarla senza averle assegnato un valore): la macchina per default può assegnare 0 oppure un intero a caso… e questo ha effetti disastrosi sul resto del programma. FATE ATTENZIONE.
- Abbiamo spostare la formula del volume prima dell'assegnamento.

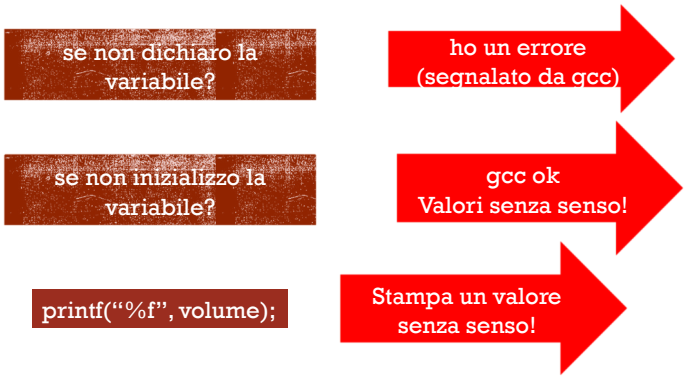
17

weight.c

```
/* Calcola il volume di un parallelepipedo di dim
 8,10,12*/

#include <stdio.h>
int main(void)
{
  int height, length, width, volume;
  height = 8;
  length = 12;
  width = 10;
  volume = height * length * width;  //formula

  //provate ad aggiungere questa printf
  printf("Dimensioni: %d,%d,%d\n", length, width, height);

  printf("Volume: %d\n", volume);


  return 0;
}
```

18

Dichiarazioni & Assegnamenti: abbiamo verificato che...



19

ESERCIZIO 1 BIS: SCRIVERE UN PROGRAMMA CHE CALCOLI IL VOLUME DI UN PARALLELEPIPEDO DI ALTEZZA, BASE E PROFONDITÀ *DATE DALL'UTENTE*. COMPILARLO. ESEGUIRLO.

1) Dammi B la base,H l'altezza, W la profondità [printf, scanf: stdio.h]
2) [il programma computa, eseguendo le istruzioni, BLACK BOX per l'utente, ignaro del «come»] Il volume è il risultato di B*H*W
3) Comunicazione del risultato all'utente:

Il volume è: … [printf]

Come si fa a "dare" i valori?

20

COME LAVORA SCANF
(FORMAT STRING SENZA CARATTERI ORDINARI)

```
scanf("%d%d%f%f", &i, &j, &x, &y);
```

```
  1
-20   .3
  -4.0e3
```

`scanf` vede il flusso di caratteri (¤ rappresenta new-line):

`••1¤-20•••.3¤•••-4.0e3¤`

`ssrsrrrsssrrssssrrrrrr` (`s` = skipped; `r` = read)

salta caratteri "**bianchi**": spazi, newline, tab

`scanf` "peeks" (sbircia) l'ultimo new-line, senza consumarlo.
Ritorneremo su questo anche quando studieremo il cap. 7…

21

weight_input.c

```
/* Calcola il volume di un parallelepipedo di dim date in
 input*/

#include <stdio.h>

int main(void)
{
  int height, length, width, volume;









  volume = height * length * width;

  printf("Dimensioni: %d,%d,%d\n", length, width, height);
  printf("Volume: %d\n", volume);
  return 0;
}
```

22

weight_input.c

```
/* Calcola il volume di un parallelepipedo di dim date in
 input*/

#include <stdio.h>

int main(void)
{
  int height, length, width, volume;


  scanf("%d", &height);

  scanf("%d", &length);

  scanf("%d", &width);

  volume = height * length * width;


  printf("Volume: %d\n", volume);
  return 0;
}
```

Eseguendo questo codice, l'utente si trova di fronte a un terminale con cursore che lampeggia e non sa cosa fare…

E' buona norma prima della scanf inserire una printf che guida l'utente nell'immissione del dato.

23

INTRODUCI ERRORI NELLA SCANF…

Abbiamo verificato che se l'utente non inserisce un intero, ma ad esempio un valore float (10.5 ad esempio), il programma termina producendo le stampe successive e un valore anomalo per volume.

La prima scanf salverebbe in height 10, si ferma sul . e da qui prosegue per la successiva scanf. Siccome la stringa di formato richiede un intero, questa scanf fallisce e non viene assegnato nulla (anzi, resta quel valore intero che per default la macchina assegna ad una variabile dichiarata ma non inizializzata).

su questo torneremo martedì.

24

weight_input.c

```
/* Calcola il volume di un parallelepipedo di dim date in
 input*/

#include <stdio.h>

int main(void)
{
  int height, length, width, volume;









  volume = height * length * width;

  printf("Dimensioni: %d,%d,%d\n", length, width, height);
  printf("Volume: %d\n", volume);
  return 0;
}
```

25

weight_input.c

```
/* Calcola il volume di un parallelepipedo di dim date in
 input*/

#include <stdio.h>

int main(void)
{
  int height, length, width, volume;

  printf("Dammi l'altezza: ");
  scanf("%d", &height);
  printf("Dammi la lunghezza: ");
  scanf("%d", &length);
  printf("Dammi la profondità: ");
  scanf("%d", &width);

  volume = height * length * width;

  printf("Dimensioni: %d,%d,%d\n", length, width, height);
  printf("Volume: %d\n", volume);
  return 0;
}
```

```
> Dimensioni: dipende…
```

26

```c
/* Calcola il volume di un parallelepipedo di dim date in input*/
#include <stdio.h>

int main(void)
{
  int height, length, width, volume;

  volume = height * length * width;

  printf("Dammi l'altezza: ");
  scanf("%d", &height);

  printf("Dammi la lunghezza: ");
  scanf("%d", &length);

  printf("Dammi la profondità: ");
  scanf("%d", &width);

  printf("Dimensioni: %d,%d,%d\n", length, width, height);
  printf("Volume: %d\n", volume);
  return 0;
}
```

Cosa non è corretto?

27

weight_input_sbagliato.c

```c
/* Calcola il volume di un parallelepipedo di dim date in input*/
#include <stdio.h>

int main(void)
{
  int height, length, width, volume, weight;

  volume = height * length * width;  /*formula ok per il
            compilatore, ma errore per l'esecuzione */

  printf("Dammi l'altezza: ");
  scanf("%d", &height);

  printf("Dammi la lunghezza: ");
  scanf("%d", &length);

  printf("Dammi la profondità: ");
  scanf("%d", &width);

  printf("Dimensioni: %d,%d,%d\n", length, width, height);
  printf("Volume: %d\n", volume);
  return 0;
}
```

28

INTRODUCI ERRORI NELLA SCANF

Provate a vedere cosa succede con

```
printf("Dimensioni: %d,%d\n", length, width, height);
```

oppure con

```
printf("Dimensioni: %d,%d\n", length);
```

è corretta? gcc segnala qualcosa?

29

PROGRAMMA: SOMMARE FRAZIONI

- Il programma `addfrac.c` chiede all'utente di inserire due frazioni e ne fornisce la frazione somma

```
Enter first fraction: 5/6
Enter second fraction: 3/4
The sum is 38/24
```

scanf???

il codice è sul libro (alla fine del capitolo 3).
Studiarlo da soli la parte relativa alla scanf.
Il resto lo vedremo insieme.

30

COME LAVORA SCANF

(FORMAT STRING CON CARATTERI ORDINARI)

```
Enter first fraction: 5/6
Enter second fraction: 3/4
The sum is 38/24
```

- Supponiamo che la stringa sia **"%d/%d"** e l'input inserito dall'utente sia •5/•96, scanf ha successo (spazi "prima" sono ignorati, saltati, condensati…)
- Se la stringa è •5•/•96 , scanf fallisce: / nella stringa non corrisponde allo spazio nell'input.
- Per consentire spazi prima del simbolo /, usa

"%d /%d"

31

CI SIAMO CONFRONTATI CON QUESTO CODICE

```
#include<stdio.h>

int main(void) {

 int d,m,a;

 printf("Inserisci una data secondo questo formato dd/mm/aaaa: ");

 scanf("%d/%d/%d", &d, &m, &a);

 printf("Data inserita: %d/%d/%d\n", d,m,a);
return 0;
}
```

Abbiamo verificato che l'utente può inserire spazi bianchi (spazio, invio, tab) prima di digitare il numero e non ci sono problemi.

Se digita 08/07/2002, verrà stampato 8/7/2002 (perché?) e quindi cambiamo la stringa di formato in questo modo per far stampare lo 0 avanti.

```
printf("Data inserita: %.2d/%.2d/%d\n", d,m,a);
```

32

CI SIAMO CONFRONTATI CON QUESTO CODICE

```
#include<stdio.h>

int main(void) {

 int d,m,a;

 printf("Inserisci una data secondo questo formato dd/mm/aaaa: ");

 scanf("%d/%d/%d", &d, &m, &a);

 printf("Data inserita: %d/%d/%d\n", d,m,a);
return 0;
}
```

se però l'utente inserisce come data 08 /07/2020, questa scanf fallisce perchè dopo l'intero deve esserci necessariamente / e non lo spazio.

Questo spazio serve alla scanf per capire che l'intero è terminato, ma poi deve esserci un match con /, che invece non c'è.

Se voglio questo, devo scrivere

```
scanf("%d /%d /%d", &d, &m, &a);
```

In tal modo posso (ma non sono obbligato) inserire spazi dopo l'intero.

33

ESERCIZIO [K. 3.3] DA FARE A CASA

Confrontate le seguenti coppie di chiamate a **scanf** e controllate se sono equivalenti o no.

```
(a) "%d"              " %d"
(b) "%d-%d-%d"        "%d -%d -%d"
(c) "%f"              "%f "
(d) "%f,%f"           "%f, %f"
```

34

SPECIFICHE DI CONVERSIONE: USO

Consideriamo il seguente codice. E' corretto?

```
#include <stdio.h>

int main(void){
 int a;
 scanf("%d\n", a);
 printf("%d", a);
 return 0;
}
```

Rispondete prima «carta e penna», poi scrivete poi il programma e verificate con gcc.
Se non è corretto, correggete prima. Poi eseguite. Cosa accade?

35

SPECIFICHE DI CONVERSIONE: USO

Consideriamo il seguente codice. E' corretto?

```
#include <stdio.h>

int main(void){
 int a;
 scanf("%d ", a);
 printf("%d", a);
 return 0;
}
```

Rispondete prima «carta e penna», poi scrivete poi il programma e verificate con gcc.
Se non è corretto, correggete prima. Poi eseguite. Cosa accade?

36

SPECIFICHE DI CONVERSIONE: USO

Consideriamo il seguente codice. E' corretto?

```
#include <stdio.h>

int main(void){
 int a;
 scanf("%d\n", &a); //cosa accade ora?
 printf("%d", a);
 return 0;
}
```

Rispondete prima «carta e penna», poi scrivete poi il programma e verificate con gcc.
Se non è corretto, correggete prima. Poi eseguite. Cosa accade?

37

ESERCIZI

1) Indicare cosa succede con scanf("%d%d", &i,&j) e l'utente inserisce 3,5.

2) Indicare cosa succede con scanf("%d, %d", &i,&j) e l'utente inserisce 34 56

38

ERRORI COMUNI

```
printf("%d %d\n", &i, &j);   /*** WRONG?? ***/
scanf("%d, %d", &i, &j);     /*** WRONG ?? **/
scanf("%d\n", &i);           /*** WRONG ?? **/
scanf("%d", i);              /*** WRONG ?? **/
```

39

3) Consideriamo il seguente frammento di codice (i numeri corrispondono alle linee di codice: è un modo sicuro per riferirsi alle istruzioni di un programma)

```
1. printf("Inserisci un intero, poi un float, infine un
   intero\n");
2. int i,k;
3. float j;
4. scanf("%d%f%d", &i,&j,&k);
5. printf("%d %f %d\n", i,j,k);
6. printf("Dammi un altro valore intero\n");
7. int z;
8. scanf("%d", &z);
9. printf("z= %d\n", z);
```

Supponiamo che in corrispondenza della printf alla linea 1, l'utente digiti `12.3 4 5` e poi batta `INVIO`. Cosa verrà stampato dalla printf alla linea 5?

Quale valore sarà assegnato alla variabile z che verrà stampato alla linea 9?

40

SPECIFICHE DI CONVERSIONE: USO

Consideriamo il seguente frammento di codice:

```
int x,y;
printf(“Inserisci un intero\n”);
scanf("%1d", &x);
printf("inserisci un altro intero\n");
scanf("%d", &y);
printf("x= %d, y=%d\n", x,y);
```

Descrivere cosa accade quando l’utente digita 23 e batte INVIO e, se si esegue nuovamente il programma, quando l’utente inserisce 5 e batte INVIO.

PROVIAMO PER CREDERE!!!

41

la scanf è insidiosa… molto… error-prone…

vi assegno esercizi che «dovete» fare carta e penna…
e **poi** «divertirvi» a vedere cosa succede con un programmino che semplicemente dichiara le variabili in questione, usa la scanf come scritta nell’esercizio.
Aggiungete una printf del contenuto delle variabili dopo le scanf. Serve per vedere se la scanf a correttamente assegnato il valore.

Eseguite poi il codice come viene scritto nell’esercizio (rispettando gli spazi o newline che ci sono) e vedete cosa viene stampato

Questo chiude i capitoli 1-2-3
(tranne qualcosa che vedremo più avanti).

42