

Lezione 4

26 Settembre 2022

1



Ricordate

- ♻ Solo fino al 13 ottobre duplico, poi solo e-learning
<http://docenti.unisa.it/020880/home>
<http://elearning.informatica.unisa.it/el-platform/>

- ♻ Consultate @studenti.unisa.it

dovete immatricolarvi!

- ♻ iscrivetevi al Team (codice pubblicato)



2



Ricordate



- o Solo fino al 13 ottobre duplico, poi solo e-learning
<http://docenti.unisa.it/020880/home>
<http://elearning.informatica.unisa.it/el-platform/>

- o Consultate @studenti.unisa.it



richiesta di accesso ai
laboratori

3

Avete fatto i «compiti a casa»???

4

Passi: dal problema alla soluzione

1. Analisi

2. Formalizzazione

3. Programmazione

4. Traduzione

5. Esecuzione

6. Verifica (testing)



```
#include <stdio.h>

int main()
{
    printf ("C Rocks!");
    return 0;
}
```

rocks.c

1

Source

You start off by creating a source file. The source file contains human-readable C code.

```
File Edit Window Help Compile
> gcc rocks.c -o rocks
>
```

2

Compile

You run your source code through a compiler. The compiler checks for errors, and once it's happy, it compiles the source code.

In Windows, this will be called rocks.exe instead of rocks.

```
1001
1110100
001010
1010111
```

rocks

3

Output

The compiler creates a new file called an *executable*. This file contains machine code, a stream of 1s and 0s that the computer understands. And that's the program you can run.

5

```
#include <stdio.h>

int main()
{
    printf ("C Rocks!");
    return 0;
}
```

rocks.c

1

Source

You start off by creating a source file. The source file contains human-readable C code.

```
File Edit Window Help Compile
> gcc rocks.c -o rocks
>
```

You run your source code through a compiler. The compiler checks for errors, and once it's happy, it compiles the source code.

In Windows, this will be called rocks.exe instead of rocks.

```
1001
1110100
001010
1010111
```

The compiler creates a new file called an *executable*. This file contains machine code, a stream of 1s and 0s that the computer understands. And that's the program you can run.

```
> ./rocks
C rocks!
>
```

Si può anche rinominare il file eseguibile:
gcc nome_file.c -o nome_file
[faremo in lab]

6

Nota:

Per introdurre i nuovi concetti per poter iniziare a lavorare,

devo spiegarvi alcune cose che faremo «dal vivo» in laboratorio [proveremo varie cose...]

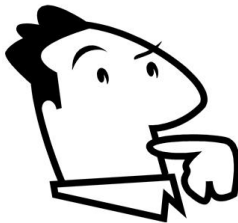
I primi capitoli 1-2-3 saranno un po' rivisitati nella presentazione, poi seguiremo precisamente il testo.

7

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.



Apro un Editor di testo e scrivo il codice



“Ogni tanto” salvo
(ricorda .c e ricordati dove lo hai salvato!)



Apro il TERMINAL,
eseguo con gcc

8

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;
}
```

9

```
/* 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;
}
```

Commenti: ...documentazione..



Attenzione ai commenti non chiusi...
E' possibile usare anche //
Q&A su cosa fa gcc con i commenti...

10

Attenzione: importante leggere queste parti del testo...



warning...

Q&A

questions and answers

11

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

#include<stdio.h> //header; iniziano con #

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

Forma generale di un programma C (semplice):

direttive
int main(void)
{
 istruzioni
}

le informazioni in stdio.h
devono essere incluse
PRIMA della compilazione

12

```

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

#include<stdio.h>//header; iniziano con #, unica linea

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

```

informazioni sulle funzioni di libreria per l'I/O

Forma generale di un programma C (semplice):

direttive

```

int main(void)
{
    istruzioni
}

```

13

```

/* 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;
}

```

Forma generale di un programma C (semplice):

direttive

```

int main(void)
{
    istruzioni
}

```

delimitano blocchi di istruzioni

14

```

/* 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;
}

```

Forma generale di un programma C (semplice):

```

direttive
int main(void)
{
    istruzioni
}

```

comandi che devono essere eseguiti.
Devono terminare con ;

15

```

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

#include<stdio.h>

int main(void) //funzione principale, MANDATORY
{
    printf("To C, or not to C: that is the question.\n");
    return 0;
}

```

Forma generale di un programma C (semplice):

```

direttive
int main(void)
{
    istruzioni
}

```

int: valore di ritorno [valore restituito dal sistema operativo al termine del programma]

16



```

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

#include<stdio.h>

int main(void) //funzione principale, MANDATORY
{
    printf("To C, or not to C: that is the question.\n");
    return 0;
}

```



si ha 1 se manca il main,
ad esempio

Forma generale di un programma C (semplice):

```

direttive
int main(void)
{
    istruzioni
}

```

int: valore di ritorno [valore restituito dal sistema operativo al termine del programma]

17

```

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

#include<stdio.h>

int main(void) //funzione principale, MANDATORY
{
    printf("To C, or not to C: that is the question.\n");
    return 0;
}

```

Forma generale di un programma C (semplice):

```

direttive
int main(void)
{
    istruzioni
}

```

int: valore di ritorno [valore restituito dal sistema operativo al termine del programma]

void: la funzione main non ha argomenti ("input")

18

```

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

#include<stdio.h>

int main(void) //funzione principale, MANDATORY
{
    printf("To C, or not to C: that is the question.\n");
    return 0;
}

```

return 0: (non è MANDATORY, warning) HA DUE COMPITI
 -fa finire il programma
 - indica che restituisce 0 (terminazione normale)

Forma generale di un programma C (semplice):

direttive
int *main*(void)
 {
 istruzioni
 }

int: valore di ritorno [valore restituito dal sistema operativo al termine del programma]

void: la funzione main non ha argomenti ("input")

19

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;
}

```

compiliamo
(gcc nomefile.c)

20

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;
}
```

Eseguiamo
(./a.out)

21

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;
}
```

Eseguiamo
(./a.out)

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

22

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;
}
```

è una specie di template che scriveremo come prima cosa quando dobbiamo scrivere un programma

23

RAGIONAMENTO... ALGORITMO

*PER IMPLEMENTARE DI COSA HO
BISOGNO?*

Come gestire l'input

Come i conti o le operazioni

Come comunicare il risultato

24

Fondamentali di C

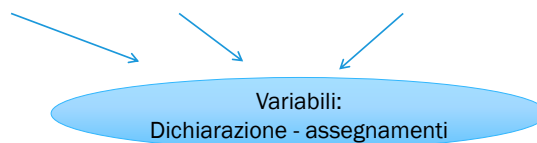
[Cap. 2-3]



25

Esercizio 1: Scrivere un programma che calcoli il volume di un parallelepipedo di altezza 8, base 10, profondità 12. [Compilarlo. Eseguirlo.]

1) Sia $B=10$ base, $H=8$ altezza, $W=12$ profondità



2) Il volume è il risultato di $B*H*W$



*Come “prendiamo” i valori?
Come comunichiamo il risultato?*

26

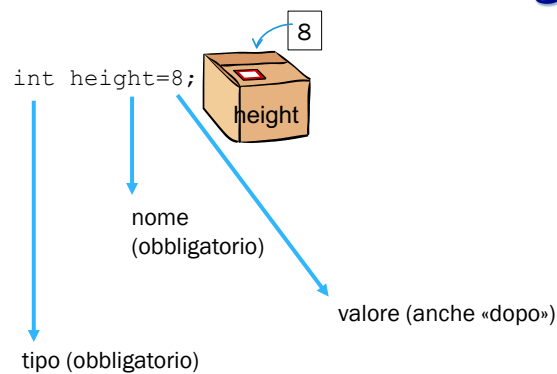
[K. Cap. 2.4]

Variabili, Assegnamenti e Tipi

- In molti algoritmi si ha bisogno di salvare i dati
- Quindi quando scriviamo il programma, dovremo conservare temporaneamente durante l'esecuzione questi valori.
- Queste locazioni di memoria sono chiamate **variabili**

27

Dichiarazioni & Assegnamenti



28

[K. Cap. 2.4]

Variabili, Tipi

- In molti algoritmi si ha bisogno di salvare i dati
- Quindi quando scriviamo il programma, dovremo conservare temporaneamente durante l'esecuzione questi valori.
- Queste locazioni di memoria sono chiamate **variabili**

importanza
del tipo

- ◆ Ogni variabile DEVE avere un tipo - **type** (ad es. `int`, `float`)
 - Il più grande `int` value è tipicamente 2,147,483,647
 - Una variabile di tipo `float` (*floating-point*) può conservare interi più grandi. anche interi con punto decimale (32.26)

29

Declarations - dichiarazioni

Prima che una variabile sia usata, deve essere dichiarata

```
int height;
float profit;
```

```
int height, length, width, volume;
float profit, loss;
```

Assignment - assegnamenti

*Ad una variabile si dà un valore (detto costante) attraverso l'assegnamento. Può anche essere **un'espressione**.*

```
height = 8;
profit = 23.45;
```

```
profit = 23.45f;
```

Q&A: perché "f"

```
profit = 23; /*ok, ma mischiare i
tipi potrebbe essere "non safe"*/
```

30



Come traduco le operazioni?

devo scrivere l'operazione e una variabile
che conserva il risultato

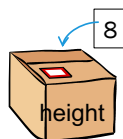


assegnamento

31

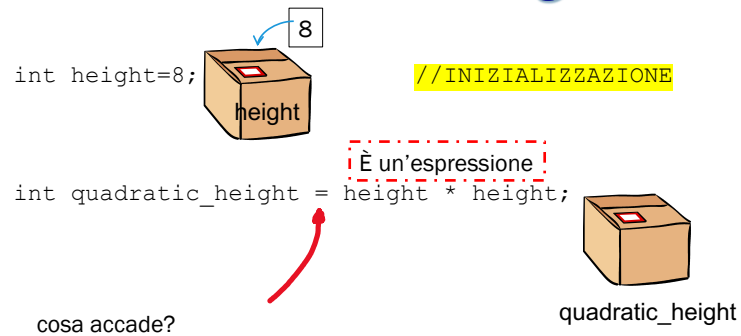
Dichiarazioni & Assegnamenti

```
int height=8;
```



32

Dichiarazioni & Assegnamenti



33

Per ora le istruzioni sono le operazioni aritmetiche...

In C esistono 5 *operatori aritmetici binari*:

- + addizione
- sottrazione
- * moltiplicazione
- / divisione [se applicate a due interi, risultato è intero!]
- % resto (o modulo), applicato a interi

34

Espressioni??? In breve e in parte...

Applico una operazione aritmetica a due (o più) variabili: in realtà la applico al loro contenuto.

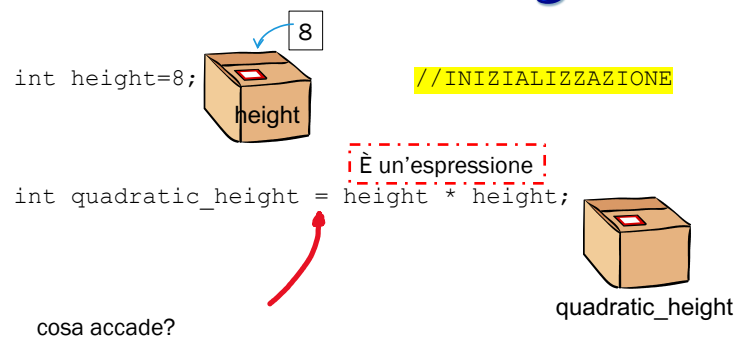
Il risultato viene:

- visualizzato (stampa), oppure
- assegnato ad un'altra variabile per fare altre operazioni successivamente.

"In parte" perchè le espressioni non sono solo quelle aritmetiche...

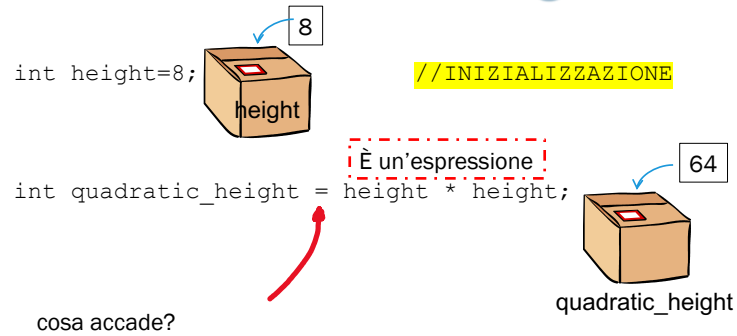
35

Dichiarazioni & Assegnamenti



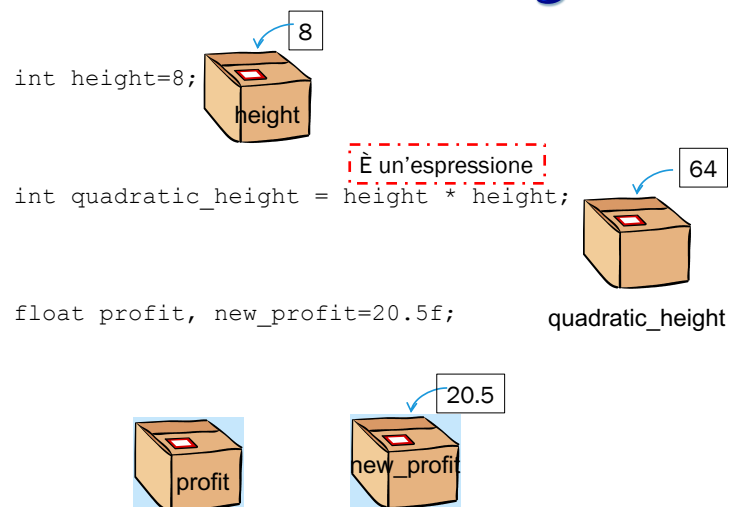
36

Dichiarazioni & Assegnamenti



37

Dichiarazioni & Assegnamenti



38

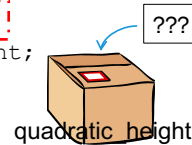
Dichiarazioni & Assegnamenti

```
int height;
```



```
int quadratic_height = height * height;
```

È un'espressione



PRIMA DELL'USO,
DEVO ASSEGNARE UN VALORE ALLA VARIABILE

39

Identifiers - identificatori

- Nomi per variabili, funzioni, macro, e altre entità.
- Può contenere lettere, cifre, underscores (separati), ma deve cominciare con una lettera [o underscore]:

Corretti: times10 get_next_char _done

Sbagliati: 10times get-next-char

- Non c'è limitazione alla lunghezza (meglio "lunghi" e "chiari")
- **C è case-sensitive: distingue tra maiuscole e minuscole...**
ATTENZIONE!

40

Keywords

- Le seguenti **keywords** non possono essere usate come identificatori:

auto	enum	restrict*	unsigned
break	extern	return	void
case	float	short	volatile
char	for	signed	while
const	goto	sizeof	_Bool*
continue	if	static	_Complex*
default	inline*	struct	_Imaginary*
do	int	switch	
double	long	typedef	
else	register	union	

*C99 only

41

Esercizio 1: Scrivere un programma che calcoli il volume di un parallelepipedo di altezza 8, base 10, profondità 12. [Compilarlo. Eseguirlo.]

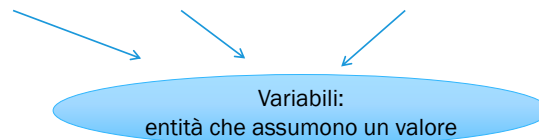
1) Sia **B=10** base, **H=8** altezza, **W=12** profondità

2) Il volume è il risultato di $B \cdot H \cdot W$

42

Esercizio 1: Scrivere un programma che calcoli il volume di un parallelepipedo di altezza 8, base 10, profondità 12. [Compilarlo. Eseguirlo.]

1) Sia $B=10$ base, $H=8$ altezza, $W=12$ profondità



2) Il volume è il risultato di $B \cdot H \cdot W$



*Come “prendiamo” i valori?
Come comunichiamo il risultato?*

43

Esercizio 1: Scrivere un programma che calcoli il volume di un parallelepipedo di altezza 8, base 10, profondità 12. Compilarlo. Eseguirlo.

1) Sia $B=10$ base, $H=8$ altezza, $W=12$ profondità



2) Il volume è il risultato di $B \cdot H \cdot W$

*I valori sono “fissi”: risolto.
Ora, come comunichiamo il risultato?*

44

weight.c

```
/* Calcola il volume di un parallelepipedo di dim  
8,10,12*/  
  
#include <stdio.h>  
int main(void)  
{  
  
    return 0;  
}
```

Dobbiamo

- 1) Dichiarare le 4 variabili
- 2) Assegnare i 3 valori
- 3) dobbiamo assegnare alla quarta il risultato del prodotto
- 4) **Stampare**

45

Dobbiamo

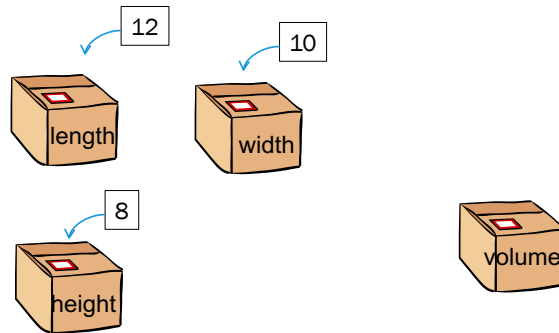
- 1) **Dichiarare le 4 variabili**
- 2) Assegnare i 3 valori
- 3) dobbiamo assegnare alla quarta il risultato del prodotto
- 4) Stampare



46

Dobbiamo

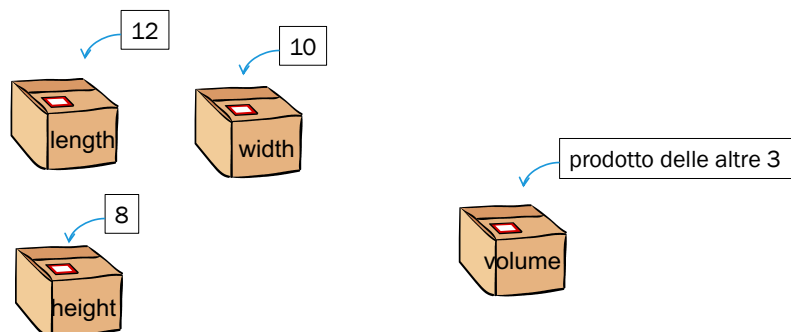
- 1) Dichiarare le 4 variabili
- 2) Assegnare i 3 valori
- 3) dobbiamo assegnare alla quarta il risultato del prodotto
- 4) Stampare



47

Dobbiamo

- 1) Dichiarare le 4 variabili
- 2) Assegnare i 3 valori
- 3) dobbiamo assegnare alla quarta il risultato del prodotto
- 4) Stampare



48

weight.c

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

#include <stdio.h>
int main(void)
{
    int height, length, width, volume;

    return 0;
}
```

49

weight.c

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

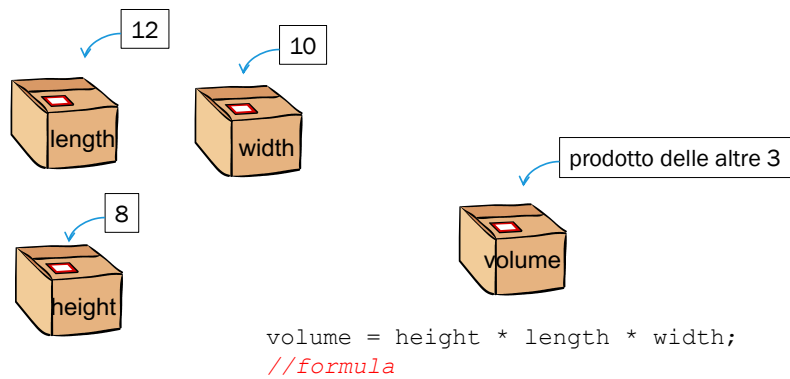
#include <stdio.h>
int main(void)
{
    int height, length, width, volume;
    height = 8;
    length = 12;
    width = 10;

    return 0;
}
```

50

Dobbiamo

- 1) Dichiarare le 4 variabili
- 2) Assegnare i 3 valori
- 3) dobbiamo assegnare alla quarta il risultato del prodotto
- 4) Stampare



51

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

    return 0;
}
```

52

Stampare espressioni

printf consente di visualizzare una qualsiasi variabile o espressione

```
printf("%d\n", volume); //stampa il valore di volume
```

53

Stampare espressioni

printf consente di visualizzare una qualsiasi variabile o espressione

```
printf("%d\n", volume); //stampa il valore di volume
```

```
printf("Dimensioni: %d,%d,%d\n", length,width,height);  
//stampa ordinatamente il valore di  
//base,altezza,prof e va a capo
```

```
> Dimensioni: 10,8,12
```

54

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

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

    return 0;
}
```

vedremo che possiamo
eliminare qualcosa...