

Il linguaggio C

I linguaggi di programmazione ad **alto livello** sono linguaggi formali (sintassi e semantica formalmente definite) però sono compatti e comprensibili. Le tipologie di linguaggi sono:

- ▶ procedurali o imperativi (C, Pascal, Fortran, ecc)
- ▶ object-oriented (Java, C++, ecc)
- ▶ funzionali (Caml, Haskel, ecc)
- ▶ logici (Prolog)

I linguaggi imperativi a cui appartiene il C sono quelli più simili come modello di calcolo al linguaggio macchina.

- ▶ Vedremo il cosiddetto **ANSI C** (standard del 1989, con successive aggiunte)
- ▶ Il primo programma C: ciao mondo

```
#include <stdio.h>
main()
  /* Stampa un messaggio sullo schermo. */
  {
    printf("Ciao mondo!\n");
  }
```



- ▶ Questo programma stampa sullo schermo una riga di testo:

Ciao mondo!

>

- ▶ Vediamo in dettaglio ogni riga del programma.

```
/* Stampa un messaggio sullo schermo. */
```

- ▶ testo racchiuso tra “/*” e “*/” è un **commento**
- ▶ i commenti servono a chi scrive o legge il programma, per renderlo più comprensibile
- ▶ il compilatore ignora i commenti
- ▶ attenzione a non dimenticare di **chiudere** i commenti con */, altrimenti tutto il resto del programma viene ignorato

main()

- ▶ è una parte presente in tutti i programmi C
- ▶ le parentesi “(” e “)” dopo main indicano che main è una **funzione**
- ▶ i programmi C sono composti da una o più funzioni, tra le quali ci **deve** essere la funzione `main`
 - ⇒ `main` è una **funzione speciale**, perché l'esecuzione del programma incomincia con l'esecuzione di `main`
- ▶ la parentesi “{” apre il **corpo** della funzione e “}” lo chiude
 - ▶ la coppia di parentesi e la parte racchiusa da esse costituiscono un **blocco**
 - ▶ il corpo della funzione contiene le istruzioni (e dichiarazioni) che costituiscono la funzione

```
printf("Ciao mondo!\n");
```

- ▶ è un'istruzione semplice (ordina al computer di eseguire un'azione) in questo caso visualizzare (stampare) sullo schermo la sequenza di caratteri tra apici
- ▶ ogni istruzione semplice deve terminare con ";"
- ▶ oltre alle istruzioni semplici, esistono anche istruzioni composte (che non devono necessariamente terminare con ";")
- ▶ la parte racchiusa in una coppia di doppi apici è una stringa (di caratteri)
- ▶ "\n" non viene visualizzato sullo schermo, ma provoca la stampa di un carattere di fine riga
 - ▶ "\" è un carattere di escape e, insieme al carattere che lo segue, assume un significato particolare (sequenza di escape)
- ▶ in realtà anche printf è una funzione, e l'istruzione di sopra è un'invocazione di funzione (le vedremo più avanti)

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

- ▶ è una **direttiva di compilazione**
- ▶ viene interpretata dal compilatore durante la compilazione
- ▶ la direttiva “**#include**” dice al compilatore di includere il contenuto di un file nel punto corrente
- ▶ **<stdio.h>** è un file che contiene i riferimenti alla libreria standard di input/output (dove è definita la funzione **printf**)
- ▶ il linguaggio C non prevede istruzioni esplicite di input/output. Queste operazioni sono definite tramite funzioni nella libreria standard di input/output.

Note:

- ▶ è importante distinguere i caratteri maiuscoli da quelli minuscoli **Main**, **MAIN**, **Printf**, **PRINTF** non andrebbero bene
- ▶ si è usata l'**indentazione** per mettere in evidenza la struttura del programma.

Alcune varianti del programma

```
#include <stdio.h>
main()
  /* Stampa un messaggio sullo schermo. */
  {
    printf("Ciao");
    printf(" mondo!\n");
  }
```

- ▶ produce lo stesso effetto del programma precedente
- ▶ la seconda invocazione di `printf` incomincia a stampare dal punto in cui aveva smesso la prima
- ▶ Cosa viene stampato se usiamo

```
printf("Ciao");
printf("mondo!\n");
```

```
printf("Ciao\n");
printf("mondo!\n");
```

Un altro programma: area di un rettangolo



```
#include <stdio.h>

main() {
    int base;
    int altezza;
    int area;

    base = 3;
    altezza = 4;
    area = base * altezza;

    printf("Area: %d\n", area);
}
```

Quando viene eseguito stampa:

```
Area: 12
>
```

Le variabili

Servono a rappresentare, nei programmi, le associazioni (modificabili) dello stato

Una variabile è caratterizzata dalle seguenti **proprietà**:

1. **nome**: serve a identificarla — esempio: `area`
2. **valore**: valore associato nello stato corrente — Esempio: `4` (può cambiare durante l'esecuzione)
3. **tipo**: specifica l'insieme dei possibili valori — Esempio: `int` (numeri interi)
4. **indirizzo**: della cella di memoria a partire dal quale è memorizzato il valore.

Nome, tipo e indirizzo **non possono cambiare** durante l'esecuzione.

Le variabili (cont.)

- ▶ Il **nome** di una variabile è un **identificatore** C
⇒ sequenza di lettere, cifre, e `_` che inizia con una lettera o con `_`
Esempio: `Numero_elementi`, `x1`, ma non `1_posto`
 - ▶ può avere lunghezza qualsiasi, ma solo i primi 31 caratteri sono significativi
 - ▶ lettere minuscole e maiuscole sono considerate distinte
- ▶ Ad ogni variabile è associata una **cella di memoria** o più celle **consecutive**, a seconda del suo tipo. Il suo **indirizzo** è quello della prima cella.
- ▶ Analogia con una scatola di scarpe etichettata in uno scaffale
 - ▶ nome ⇒ etichetta
 - ▶ valore ⇒ scarpa che c'è nella scatola
 - ▶ tipo ⇒ capienza (che tipo di scarpe ci metto dentro)
 - ▶ indirizzo ⇒ posizione nello scaffale (la scatola è incollata)

N.B.

- ▶ non tutte le variabili sono denotate da un identificatore
- ▶ non tutti gli identificatori sono identificatori di variabile (ad es. funzioni, tipi, parole riservate, ...)

Area del rettangolo

- ▶ `int base;` — è una **dichiarazione di variabile**
 - ▶ viene creata la scatola e incollata allo scaffale
 - ▶ ha **tipo** `int` \implies può contenere interi
 - ▶ ha **nome** `base`
 - ▶ ha un **indirizzo** (posizione nello scaffale), che è quello della cella di memoria associata alla variabile
 - ▶ ha un **valore iniziale**, che però non è significativo (è casuale)
 - \implies la scatola viene creata piena, però con una scarpa scelta a caso, ovvero
 - \implies l'associazione nello stato è del tipo `nome \rightsquigarrow ?`
- ▶ `int altezza;`
`int area;`
 - \implies come per `base`

Variabili numeriche

Variabili **intere**

- ▶ per dichiarare variabili intere si può usare il tipo `int`
- ▶ i valori di tipo `int` sono interi il cui valore massimo e minimo dipende dal compilatore che si utilizza. **Esempio:** per il compilatore gcc (usato in ambiente Unix) $+/- 2147483648$
- ▶ riprenderemo il discorso quando parleremo della rappresentazione.
- ▶ in C esistono altri tipi per variabili intere (`short`, `long`) — li vedremo più avanti

Variabili **razionali**

- ▶ per dichiarare variabili razionali si può usare il tipo `float`
Esempio: `float temperatura;`

Area del rettangolo

```
base = 3;
```

è un'istruzione di assegnamento

- ▶ in C l'operatore di assegnamento è denotato dal simbolo “=”
- ▶ come già sappiamo, l'effetto è di **modificare** una associazione nello stato
 - ⇒ in questo caso il valore 3 viene associato a **base**, come?
 - ⇒ il nuovo valore viene scritto nella spazio associato alla variabile
- ▶ a questo punto la variabile **base** ha un valore significativo
 - ⇒ da **base** ↔ ? a **base** ↔ 3

```
altezza = 4; ⇒ come sopra
```

```
area = base * altezza;
```

a destra di “=” possono comparire **espressioni** ⇒ il valore assegnato è quello dell'espressione calcolata nello stato corrente

- ▶ una variabile all'interno di una espressione **sta per** il valore ad essa associato in quel momento

Nota: **operatori aritmetici** tra interi del C +, -, *, /, %, ...

Area del rettangolo

```
printf("Area: %d\n", area);
```

- ▶ è un'istruzione di **stampa**
- ▶ il primo argomento è la **stringa di formato** che può contenere **specificatori di formato**
- ▶ lo specificatore di formato `%d` indica che deve essere stampato un intero in notazione decimale (`d` per decimal)
- ▶ ad ogni specificatore di formato nella stringa deve corrispondere un valore che deve seguire la stringa di formato tra gli argomenti di `printf`

Esempio: `printf("%d%d...%d", i1, i2, ..., in);`

- ▶ nel caso di `printf("Ciao mondo!\n");` la stringa di formato non conteneva specificatori e quindi non vi erano altri argomenti.