La programmazione nel linguaggio C

Introduzione ai programmi C

Vedremo il cosidetto **ANSI C** (standard del 1989, con piccole aggiunte del 1994)

Il primo programma C: ciao mondo

File: `base/ciao.c`

```
#include <stdio.h>
int main(void)
  /* Stampa un messaggio sullo schermo. */
{
  printf("Ciao mondo!\n");
  return 0;
}
```

Questo programma stampa sullo schermo una riga di testo:

```
Ciao mondo!
#
```

(“#” denota la posizione del cursore)

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 `*/`

`int main(void)`

- è una parte presente in tutti i programmi C
- le parentesi "`(`" e "`)`" dopo main indicano che main è una **funzione**
- i programmi C contengono 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`
- le funzioni C (come quelle matematiche) prendono un insieme (eventualmente vuoto) di **argomenti** e restituiscono un **valore** (oppure nulla)
  - `void` specifica che `main` non prende alcun argomento
  - `int` specifica che il valore restituito da `main` è di **tipo** intero

  Vedremo più avanti la nozione di "tipo" e i tipi del C.
- la parentesi graffa "`{`" 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** "`;`"
- vedremo più avanti che, 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'**attivazione** di funzione (le vedremo più avanti)

`return 0;`

- se usato nella funzione `main` fa terminare l'esecuzione del programma
- 0 indica che il programma è terminato con successo
- vedremo più avanti qual'è l'effetto dell'istruzione `return` quando viene usata in altre funzioni

`#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`)

Note:

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

Alcune varianti del programma `ciao.c`

```
#include <stdio.h>
int main(void)
{
  printf("Ciao");
  printf(" mondo!\n");
  return 0;
}
```

- produce lo stesso effetto del programma precedente
- la seconda `printf` incomincia a stampare dal punto in cui aveva smesso la prima

Cosa viene stampato se usiamo

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

e se usiamo

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


Un altro semplice programma: area di un rettangolo

File: `base/arearet1.c`

```
#include <stdio.h>
int main(void)
{
  int base;    int altezza;    int area;

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

  printf("Area: %d\n", area);
  return 0;
}  /* main */
```

Quando viene eseguito stampa:

```
Area: 21
#
```


Le variabili: servono a denotare i dati all'interno dei programmi.

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

1. **nome**: serve a identificarla — *Esempio:* `altezza`

   È un **identificatore** C: sequenza di lettere, cifre, e "_" che comincia con una lettera o con "_" (non con una cifra)
   - può avere lunghezza qualsiasi, ma solo i primi 31 caratteri sono significativi (ANSI)
   - lettere minuscole e maiuscole sono considerate distinte
2. **tipo**: specifica il tipo di dato che può memorizzare
   Esempio: `int` (può memorizzare interi)
3. **indirizzo**: della cella di memoria che contiene il dato denotato;

   Ad ogni variabile è associata una **cella di memoria** (o più celle di memoria consecutive, a seconda del tipo).
4. **valore**: dato che la variabile denota in un certo istante dell'esecuzione
   Esempio: 4

   Può cambiare durante l'esecuzione.

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

Analogia con una scatola di scarpe etichettata in uno scaffale

- nome $\Longrightarrow$ etichetta
- tipo $\Longrightarrow$ capienza (che tipo di scarpe ci metto dentro)
- indirizzo $\Longrightarrow$ posizione nello scaffale (la scatola è incollata)
- valore $\Longrightarrow$ scarpa che c'è nella scatola

N.B.

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

Ritorniamo al programma per l'area del rettangolo

`int altezza;`

è una **dichiarazione di variabile**

- viene creata la scatola e incollata allo scaffale
- ha **tipo** `int` $\Longrightarrow$ può contenere interi
- ha **nome** `altezza`
- 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) — la scatola viene creata piena, però con una scarpa scelta a caso

```
int base;
int area;
```

- come per `altezza`

Variabili **intere**

- per dichiarare variabili intere si può usare il tipo `int`
- valori di tipo `int` sono rappresentati in C con almeno 16 bit
- il numero effettivo di bit dipende dal compilatore
  Esempio: 32 bit per il compilatore gcc (usato in ambiente Unix)
- in C esistono altri tipi per variabili intere (`short`, `long`) — li vedremo più avanti

Variabili **reali**

- per dichiarare variabili reali si può usare il tipo `float`
  Esempio: `float temperatura`
- per immettere un reale si può usare la notazione con il punto decimale
- specificatore di formato: `%g`

`base = 3;`

è un'**istruzione di assegnazione**

- "`=`" è l'**operatore di assegnazione**
- il suo effetto è quello di **assegnare** il valore a destra di "`=`" (in questo caso 3) alla variabile a sinistra (in questo caso `base`)
  $\Longrightarrow$ il valore viene scritto nella cella di memoria associata alla variabile
- a questo punto la variabile `base` ha un valore significativo

```
altezza = 4;
area = base * altezza;
```

è un'istruzione di assegnazione, in cui a destra di "`=`" abbiamo un'**espressione**

- vengono presi i valori di `base` (3) e `altezza` (4) e viene calcolato il loro prodotto (12)
- tale valore viene assegnato alla variabile `area`

Nota: il C mette disposizione gli **operatori aritmetici** tra interi: `+`, `-`, `*`, `/`, ...

`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`

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


Area di un rettangolo di dimensioni lette da tastiera

File: `base/arearet2.c`

```
#include <stdio.h>

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

  printf("Immetti base del rettangolo e premi INVIO\n");
  scanf("%d", &base);
  printf("Immetti altezza del rettangolo e premi INVIO\n");
  scanf("%d", &altezza);

  area = base * altezza;

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

  return 0;
}  /* main */
```

`int base, altezza, area;`

- posso dichiarare contemporaneamente più variabili **dello stesso tipo**
- il tipo viene specificato un'unica volta
- le variabili devono essere separate con una virgola "`,`"
- la dichiarazione deve essere terminata con "`;`"

Quindi, la **sintassi** di una dichiarazione di variabili è

```
tipo variabile-1, variabile-2, ..., variabile-n;
```

In generale, la **sintassi** di un linguaggio di programmazione specifica le regole per la scrittura corretta di un programma e delle sue parti.

In informatica vengono usati dei formalismi appositi per specificare la sintassi dei linguaggi di programmazione. Noi lo faremo attraverso esempi significativi.

La **semantica** di un linguaggio di programmazione specifica invece qual'è il significato dei diversi costrutti di un programma.

`scanf("%d", &base);`

- `scanf` è la funzione duale di `printf`
- legge da input (tastiera) un valore intero e lo assegna alla variabile `base`
- `"%d"` è la **stringa di controllo del formato** (in questo caso viene letto un intero in formato decimale)
- "`&`" è l'**operatore di indirizzo**
  - `&base` indica (l'indirizzo del)la locazione di memoria associata a `base`
  - la funzione `scanf` utilizza tale locazione per metterci il valore letto da tastiera
- quando viene eseguita `scanf` il programma si mette in attesa che l'utente immetta un valore
- quando l'utente digita *Invio*
  (d) la sequenza di caratteri immessa viene convertita in un intero e
  (d) l'intero ottenuto viene assegnato alla variabile `base` (viene cioè scritto nella cella di memoria il cui indirizzo è stato passato a `scanf`)
- N.B. il precedente valore della variabile `base` va perduto

Cosa appare **sullo schermo**:

```
Immetti base del rettangolo e premi INVIO
# => 5^
Immetti altezza del rettangolo e premi INVIO
# => 4^
Area: 20
#
```

(quello che segue "=>" è l'input dell'utente
"^" denota la pressione del tasto *Invio*)

Esercizio: Scrivere un programma che legge da tastiera un valore (reale) in Euro e stampa il controvalore in Lire.

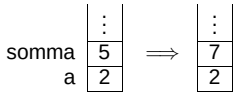
Osservazioni sull'istruzione di assegnazione

Nell'istruzione `x = e` viene

1. prima valutato il valore dell'espressione `e` a destra di "=" (usando i valori correnti delle variabili);
2. poi tale valore viene assegnato alla variabile `x` a sinistra di "=".

Esempio:

```
somma = 5;
a = 2;
somma = somma + a;
```



Esempio:

```
int a, b;

a = 2;
b = 3;
a = b;
a = a + b;
b = a + b;
```

a	b
?	?
2	?
2	3
3	3
6	3
6	9

A sinistra di "=" ci deve essere una **variabile** (ci vuole una locazione di memoria in cui scrivere il valore).

Esempio: Quali istruzioni sono corrette e quali no?

`a = a;`	SI corretta, ma ha effetto nullo
`a = 2 * a;`	SI corretta
`5 = a;`	NO, quale è la locazione di memoria
`5 = 6;`	NO, quale è la locazione di memoria nella quale scrivere il valore di `a`
`a + b = c;`	NO, `a+b` non è una variabile

Esempio: **Scambio del valore** di due variabili: prima:

dopo:

Il seguente codice non funziona (si perde il valore di `a`): `a = b; b = a;`

$\Longrightarrow$ prima di eseguire `a = b` bisogna copiare il valore di `a` in una **variabile temporanea** per poterla poi assegnare a `b`:

```
temp = a;
a = b;
b = temp;
```

a	b	temp
5	8	?
5	8	5
8	8	5
8	5	5

Gli operatori aritmetici del C

- in ordine di priorità:
  - `-` unario — priorità alta
  - `*` (moltiplicazione), `/` (divisione), `%` (modulo)
  - `+` (somma), `-` (sottrazione) — priorità bassa
- la **moltiplicazione** "`*`" va denotata esplicitamente
- "`/`" tra due interi indica la **divisione intera**

  Esempio: $24/6 = 4$ $\quad -24/6 = -4$
  $25/6 = 4$ $\quad -25/6 = -4$
- "`%`" può essere usato solo con operandi interi
- le espressioni vengono valutate **da sinistra a destra** tenendo conto delle **priorità degli operatori** (esattamente come in algebra)

  Esempio: `2 + 3 * 4` vale 14 (e non 20)

  Si possono sempre usare le **parentesi** per imporre un certo ordine di valutazione.

  Esempio: `(2 + 3) * 4` vale 20

Esercizio: Scrivere un programma che legge due interi e stampa quoziente e resto della divisione.

Es.: con 25 e 6 deve stampare "`25:6 = 4 con resto di 1`"

Esercizio: Leggere i coefficienti a, b, c di un'equazione di secondo grado

$$a \cdot x^2 + b \cdot x + c = 0$$

che si suppone essere a discriminante nonnegativo (ovvero vale che $b^2 - 4 \cdot a \cdot c \geq 0$) e calcolare gli zeri dell'equazione.