Programmazione 1

Lezione 1 – Parte 3

Vincenzo Marra

vincenzo.marra@unimi.it

Dipartimento di Matematica *Federigo Enriques*
Università degli Studi di Milano

28 febbraio 2018

Struttura del sorgente

Un programma in C è costituito da uno o più file sorgente che contengono il testo del programma, detto codice sorgente, o codice, o sorgente.

I file sorgente hanno un nome della forma

`nomedelfile.c`

L'estensione `.c` indica che si tratta, appunto, di un file contenente codice sorgente scritto nel linguaggio C.

Per tutto o quasi tutto il corso assumeremo che l'intero programma sia contenuto in un unico file sorgente.

Struttura del sorgente

Concettualmente, un programma in C è fatto da *funzioni* e *variabili*. Le prime codificano gli *algoritmi*, le seconde i *dati* elaborati dagli algoritmi.

Un file sorgente è costituito dagli elementi sintattici seguenti.

Struttura del sorgente

Concettualmente, un programma in C è fatto da *funzioni* e *variabili*. Le prime codificano gli *algoritmi*, le seconde i *dati* elaborati dagli algoritmi.

Un file sorgente è costituito dagli elementi sintattici seguenti.

- Direttive per il preprocessore. Ne parleremo più avanti.

Struttura del sorgente

Concettualmente, un programma in C è fatto da *funzioni* e *variabili*. Le prime codificano gli *algoritmi*, le seconde i *dati* elaborati dagli algoritmi.

Un file sorgente è costituito dagli elementi sintattici seguenti.

- Direttive per il preprocessore. Ne parleremo più avanti.
- Funzioni. Unità di codice che svolgono uno specifico compito. Accettano dei dati in ingresso, e restituiscono dei dati in uscita.

Struttura del sorgente

Concettualmente, un programma in C è fatto da *funzioni* e *variabili*. Le prime codificano gli *algoritmi*, le seconde i *dati* elaborati dagli algoritmi.

Un file sorgente è costituito dagli elementi sintattici seguenti.

- Direttive per il preprocessore. Ne parleremo più avanti.
- Funzioni. Unità di codice che svolgono uno specifico compito. Accettano dei dati in ingresso, e restituiscono dei dati in uscita.
- Variabili. Contengono i dati da elaborare.

Struttura del sorgente

Concettualmente, un programma in C è fatto da *funzioni* e *variabili*. Le prime codificano gli *algoritmi*, le seconde i *dati* elaborati dagli algoritmi.

Un file sorgente è costituito dagli elementi sintattici seguenti.

- Direttive per il preprocessore. Ne parleremo più avanti.
- Funzioni. Unità di codice che svolgono uno specifico compito. Accettano dei dati in ingresso, e restituiscono dei dati in uscita.
- Variabili. Contengono i dati da elaborare.
- Istruzioni. (All'interno delle funzioni.) Specificano come elaborare i dati.

Struttura del sorgente

Concettualmente, un programma in C è fatto da *funzioni* e *variabili*. Le prime codificano gli *algoritmi*, le seconde i *dati* elaborati dagli algoritmi.

Un file sorgente è costituito dagli elementi sintattici seguenti.

- Direttive per il preprocessore. Ne parleremo più avanti.
- Funzioni. Unità di codice che svolgono uno specifico compito. Accettano dei dati in ingresso, e restituiscono dei dati in uscita.
- Variabili. Contengono i dati da elaborare.
- Istruzioni. (All'interno delle funzioni.) Specificano come elaborare i dati.
- Commenti. Sono righe di codice ignorate dal compilatore, che servono a rendere il codice leggibile ad un umano.

Il primo programma in C

I programmi implementano delle specifiche. Il primo programma in C che scriveremo ha la specifica seguente.

Specifica: CiaoMondo

Input: Nulla.
Output: La stringa `Ciao Mondo`.
Descrizione: Scrivere un programma in C che produca in uscita — ossia, visualizzi sullo schermo del computer — la stringa di caratteri

```
Ciao Mondo.
```

seguita da un ritorno a capo, e quindi termini.

Scriveremo il codice all'interno dell'unico file sorgente `CiaoMondo.c`.

Il primo programma in C

CiaoMondo.c

```c
/* Il mio primo programma in C */

#include <stdio.h>

int main(void)
{
        printf("Ciao mondo.\n");
        return 0;
}
```

Il primo programma in C

CiaoMondo.c

```
1  /* Il mio primo programma in C */
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7          printf("Ciao mondo.\n");
8          return 0;
9  }
```

```
1  /* Il mio primo programma in C */
```

Questo è un commento; tutto ciò che compare fra `/*` e `*/` è ignorato dal compilatore.

Il primo programma in C

CiaoMondo.c

```c
/* Il mio primo programma in C */

#include <stdio.h>

int main(void)
{
        printf("Ciao mondo.\n");
        return 0;
}
```

```c
// Un commento su riga singola
```

I commenti su una sola riga si possono anche scrivere in questo modo. (Solo dallo standard C99 in poi.)

Il primo programma in C

CiaoMondo.c

```
1  /* Il mio primo programma in C */
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7          printf("Ciao mondo.\n");
8          return 0;
9  }
```

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

Questa riga è una direttiva per il preprocessore: include nel codice sorgente informazioni sulla libreria standard.

Il primo programma in C

CiaoMondo.c

```
1  /* Il mio primo programma in C */
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7          printf("Ciao mondo.\n");
8          return 0;
9  }
```

```
5  int main(void)
```

Questa riga definisce la funzione `main`, da cui, per convenzione, comincia l'esecuzione del programma. Essa non riceve alcun argomento dal chiamante (`void`) e restituisce un intero (`int`).

Il primo programma in C

CiaoMondo.c

```
1  /* Il mio primo programma in C */
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7          printf("Ciao mondo.\n");
8          return 0;
9  }
```

```
5  int main(void)
```

Il corpo della funzione `main`, che ne costituisce la implementazione, è racchiuso fra le parentesi graffe { e }.

Il primo programma in C

CiaoMondo.c

```
1  /* Il mio primo programma in C */
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7          printf("Ciao mondo.\n");
8          return 0;
9  }
```

```
7          printf("Ciao mondo.\n");
```

Questa riga visualizza sullo schermo la frase (o la "stringa di caratteri") `Ciao mondo` e va a capo (`\n`).

Il primo programma in C

CiaoMondo.c

```
1  /* Il mio primo programma in C */
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7          printf("Ciao mondo.\n");
8          return 0;
9  }
```

```
7          printf("Ciao mondo.\n");
```

La funzione `main` chiama la funzione `printf` della libreria standard per visualizzare la stringa di caratteri.

Il primo programma in C

CiaoMondo.c

```
1  /* Il mio primo programma in C */
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7          printf("Ciao mondo.\n");
8          return 0;
9  }
```

```
7          printf("Ciao mondo.\n");
```

La funzione `printf` serve a stampare stringhe di caratteri sul terminale. Permette quindi di produrre dati in uscita, o output.

Il primo programma in C

CiaoMondo.c

```
1  /* Il mio primo programma in C */
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7          printf("Ciao mondo.\n");
8          return 0;
9  }
```

```
7          printf("Ciao mondo.\n");
```

La funzione `printf` è definita nel file di intestazione (in inglese *header*, da cui il suffisso `.h`) del linguaggio C denominato `stdio.h`, per `standard input and output`.

Il primo programma in C

CiaoMondo.c

```
1  /* Il mio primo programma in C */
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7          printf("Ciao mondo.\n");
8          return 0;
9  }
```

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

Per usarla occorre quindi includere il file di intestazione nel programma, con questa direttiva per il preprocessore.

Il primo programma in C

CiaoMondo.c

```
1  /* Il mio primo programma in C */
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7          printf("Ciao mondo.\n");
8          return 0;
9  }
```

```
7          printf("Ciao mondo.\n");
```

Si noti che l'invocazione di `printf` è terminata da un punto e virgola (`;`).

Il primo programma in C

CiaoMondo.c

```
1  /* Il mio primo programma in C */
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7          printf("Ciao mondo.\n");
8          return 0;
9  }
```

```
8          return 0;
```

Restituisce il valore 0; indica al chiamante che l'esecuzione è terminata correttamente. Anche qui ; indica la terminazione dell'istruzione.

Compilare il primo programma in C

Per compilare il programma, si invoca dal terminale il compilatore C — che è `gcc` nell'implementazione GNU/LINUX — seguito dal nome del file sorgente:

```
gcc CiaoMondo.c
```

(Occorre premere invio per eseguire il comando dal terminale. Si assume qui che il file `CiaoMondo.c` si trovi nella directory corrente.)

Compilare il primo programma in C

Per compilare il programma, si invoca dal terminale il compilatore C — che è `gcc` nell'implementazione GNU/LINUX — seguito dal nome del file sorgente:

```
gcc CiaoMondo.c
```

(Occorre premere invio per eseguire il comando dal terminale. Si assume qui che il file `CiaoMondo.c` si trovi nella directory corrente.)

La compilazione, se termina senza errori, non visualizza alcun messaggio. Produce però nella directory corrente il file eseguibile o codice eseguibile, o semplicemente l'eseguibile:

```
a.out
```

Per lanciare il programma, si usa `./a.out`.

Compilare il primo programma in C

Se si preferisce dare un nome diverso al file eseguibile, lo si può fare così:

```
gcc -o CiaoMondo CiaoMondo.c
```

Compilare il primo programma in C

Se si preferisce dare un nome diverso al file eseguibile, lo si può fare così:

```
gcc -o CiaoMondo CiaoMondo.c
```

In questo caso la compilazione produce nella directory corrente il file oggetto:

```
CiaoMondo
```

Compilare il primo programma in C

Se si preferisce dare un nome diverso al file eseguibile, lo si può fare così:

```
gcc -o CiaoMondo CiaoMondo.c
```

In questo caso la compilazione produce nella directory corrente il file oggetto:

```
CiaoMondo
```

Nota Bene.

Nell'invocazione precedente, il compilatore sovrascriverà senza ulteriore preavviso l'eventuale file `CiaoMondo` già presente nella directory corrente, sostituendolo col codice eseguibile prodotto dalla compilazione.

Il ciclo Composizione-Compilazione-Esecuzione

(*Esemplificazione al calcolatore del ciclo composizione-compilazione-esecuzione.*)

Breve Predica Finale

La programmazione si impara studiando la teoria, ma soprattutto programmando!