CORSO DI PROGRAMMAZIONE
A.A. 2014-15

Dispensa 1

Laboratorio

Dott. Mirko Ravaioli
e-mail: mirko.ravaioli@unibo.it

1.1 Breve Storia del Linguaggio C

Il Linguaggio di programmazione C è stato creato nel 1972 da Dennis Ritchie nei laboratori della Bell Telephone, con lo scopo di realizzare il sistema operativo Unix. Il C si diffuse rapidamente anche al di fuori dei laboratori della Bell e presto diverse aziende iniziarono a utilizzare versioni proprietarie di questo linguaggio, e nacquero diverse sottili differenze di implementazione che causano ancora oggi notevoli problemi agli sviluppatori.

Per ovviare a questo problema nel 1983 l'istituto statunitense degli standard ANSI (American National Standards Institute) ha formato un comitato con l'obiettivo di produrre una definizione del C chiara e indipendente dalla macchina, capace però di preservare lo spirito originale, quindi in modo di definire uno standard per il C che da allora si chiamò ANSI C. Tranne poche eccezioni, tutti i compilatori moderni si conformano allo standard.

Il C è considerato un linguaggio di "basso livello", questa definizione non è peggiorativa ma significa semplicemente che il C adopera lo stesso tipo di oggetti usati da molti elaboratori, come caratteri, numeri e indirizzi, che possono essere combinati tramite gli operatori aritmetici e logici di cui si servono le macchine reali.

Il C permette solo il controllo di un singolo flusso di computazione: condizioni, cicli, raggruppamenti e sottoprogrammi, ma non multiprogrammazione parallele, sincronizzazione o co-rutine.

1.2 Prima di iniziare a programmare

Quando si cerca di risolvere un problema è bene seguire una cerca metodologia:

- definire precisamente il problema, altrimenti non sarà possibile individuare la soluzione
- ideare un piano risolutivo (algoritmo risolutivo)
- implementare l'algoritmo (metterlo in pratica)
- controllare i risultati in modo da verificare se il problema è stato risolto o meno.

1.3 Ciclo di sviluppo del programma

Il ciclo di sviluppo di un programma si divide in diverse parti:

- **Creazione del Codice Sorgente**: creare attraverso un editor di testo un file contenente una serie di istruzioni o comandi impiegati che combinati in una certa logica per istruire il computer in modo che esegua i compiti desiderati. Ad esempio una riga di codice sorgente C:

  ```
  printf("Forza Cesena!");
  ```

  Questa istruzione ordina al computer di stampare a video la scritta: "Forza Cesena!".

  I file che contengono il sorgente di un programma scritto in C hanno estensione .c, ad esempio:

  ```
  pippo.c
  ```

- **Creazione del Programma Sorgente**: il computer non è in grado di comprendere direttamente quello che l'utente indica attraverso il sorgente C, ma necessita di istruzioni binarie o digitali in un particolare formato detto **Linguaggio Macchina**. Quindi prima che il programma possa funzionare

su un computer, deve essere tradotto nel formato sorgente al linguaggio macchina. Questa traduzione è effettuata da un programma chiamato **compilatore** che partendo dal sorgente produce un nuovo file su disco contenente le istruzioni in linguaggio macchina che corrispondono ai comandi impartiti in C. Queste istruzioni vengono chiamate **codice oggetto** e il file creato sul disco viene chiamato **file oggetto**. Il file oggetto ha estensione .obj.

- **Creazione del File Eseguibile con il Linking**: il linguaggio C è costituito da un insieme di librerie che contengono il codice oggetto (ovvero pre-compilato) delle funzioni predefinite. Una funzione predefinita contiene codice C già scritto e fornito in forma pronta per l'uso con il compilatore stesso. Ad esempio la funzione *printf()* utilizzata nell'esempio precedente è una funzione di libreria. Una funzione potrebbe essere vista come un "robot da cucina" che esegue determinate operazioni ogni volta che lo si attiva chiamandolo per nome. Queste particolari funzioni svolgono compiti frequenti come la visualizzazione di informazioni sullo schermo o la lettura di dati da tastiera. Se il proprio programma contiene alcune di queste funzioni (difficilmente si può scrivere un programma di qualche utilità senza utilizzarne almeno una), il file oggetto prodotto a partire dal sorgente deve essere combinato con quello delle funzioni di libreria per generare un programma eseguibile. Questo processo è detto **Linking** ed è portato a termine da un programma chiamato **Linker**.

- **Completamento del ciclo di sviluppo**: una volta compilato e linkato il proprio programma è possibile eseguirlo. Se dall'esecuzione del programma non si ottengono i risultati aspettati bisogna tornare indietro al primo passo e individuare gli errori.

1.4 Primi passi

Proviamo a scrivere il primo programma che stampi a video la scritta: "Forza Cesena!".

Per risolvere questo problema dobbiamo essere in grado di redigere il testo del programma, compilarlo con successo. caricarlo, eseguirlo e scoprire dove è finito il risultato. Questi sono meccanismi essenziali per capire non solo come programmare con il linguaggio C ma con qualsiasi linguaggio di programmazione.

Il programma che risolve l'esercizio è il seguente:

```
#include <stdio.h>

int main()
{
      printf("Forza Cesena!\n");

      return 0;
}
```

Il programma una volta compilato ed eseguito (salvo errori!!) darà come risultato:

```
Forza Cesena!
```

Un programma in C di qualunque lunghezza consiste di funzioni e variabili. Una funzione consiste in un insieme di istruzioni o comandi che specificano quali operazioni devono essere effettuate, mentre le variabili memorizzano i valori usati durante l'esecuzione del programma. Nell'esempio precedente viene definita la funzione *main()*, di norma si è liberi di assegnare qualsiasi nome alle funzioni ma il caso di "main" è particolare in quanto tutti i programmi scritti in C inizieranno con la chiamata alla funzione *main()*. Quindi ogni programma dovrà avere una funzione *main*.

Per svolgere il proprio lavoro la funzione *main* chiamerà altre funzioni in aiuto, alcune scritte dal programmatore altre presenti nelle librerie cui si ha accesso. La prima riga del programma:

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

dice al compilatore di includere le informazioni sulla libreria standard per l'input/output. Tutte le istruzioni di inclusione delle librerie devono comparire all'inizio del programma.

Tutte le funzioni hanno un inizio ed una fine, per individuare all'interno del codice dove una funzione inizia e finisce si utilizzano rispettivamente due marcatori: { (parentesi graffa aperta) e } (parentesi graffa chiusa). (premere pulsanti ALT GR + SHIFT + QUADRA APERTA per parentesi graffa aperta e premere ALT GR + SHIFT + QUADRA CHIUSA per parentesi graffa chiusa). All'interno delle graffe è presente il corpo della funzione (insieme di istruzioni eseguite alla chiamata della funzione).

Nel esempio precedente all'interno del corpo della funzione *main()* troviamo 2 righe di codice, in paricolare la prima consiste nella chiamata della funzione *printf()*

```
printf("Forza Cesena!\n");
```

Una funzione è chiamata attraverso il suo nome, seguito da un elenco degli argomenti racchiuso tra parentesi tonde. Gli argomenti sono il metodo di comunicazione tra le varie funzioni in modo che la funzione chiamante riesca a fornire alla funzione chiamata un insieme di valori detti argomenti. Nel nostro esempio la funzione si chiama *printf* e l'argomento passato è *"Forza Cesena!"*. Si tratta di una funzione che produce (in gergo si dice anche stampa) a video o in uscita dei dati, in questo caso in particolare l'effetto finale è quello di visualizzare a schermo la scritta.

Una successione di caratteri è comunemente detta **stringa di caratteri** ed è individuata da un doppio apice in apertura e in chiusura.

La sequenza \n che compare nella stringa è una notazione del C per accedere alla *newline*, che ha effetto di spostare in avanti di una riga i dati in uscita (in poche parole di andare a capo!). Senza il \n non si va a capo. La funzione *printf()* non fornisce in automatico l'andata a capo. Quindi il programma di prima poteva essere scritto anche nel seguente modo:

```
#include <stdio.h>

int main()
{
      printf("Forza ");
      printf("Cesena!");
      printf("\n");

      return 0;
}
```

L'istruzione **return** è molto importante in quanto consente di restituire un parametro alla funzione chiamante. Nel nostro caso il chiamate è il sistema operativo, il nostro programma terminate le proprie operazioni restituisce il valore 0 (zero) informando il sistema operativo che tutte le operazioni si cono svolte senza errori.

1.4.1 Le istruzioni

Da notare che nel programma ogni istruzione presente all'interno della funzione *main()* termina con un punto e virgola.

Un'istruzione è un comando completo che indica al computer di eseguire un particolare compito. Le istruzioni in C terminano sempre con un punto e virgola (tranne il caso delle direttive del preprocessore #define o #include).

Quindi scrivere

```
x = 1 + 3;
```

equivale a scrivere:

```
x =
1
+
3;
```

Mentre scrivere

```
printf("ciao a tutti");
```

non equivale a

```
printf(" Ciao
a tutti");
```

in quanto genera un errore. Mentre scrivere

```
printf(" Ciao\
a tutti");
```

non genera errore in quanto per andare a capo con una stringa è possibile utilizzare il carattere (\) di barra inversa.

1.4.2 La funzione printf()

La stringa di formato della funzione *printf()* specifica la modalità di formattazione dell'output e ha tre componenti possibili:

- Il *testo letterale*, che viene visualizzato esattamente come viene introdotto nella stringa
- Un *indicatore di conversione* formato da un segno di percentuale(%) seguito da un carattere.
- Una *sequenza di escape* che consente di produrre comandi di formattazione particolari. Le sequenze di escare sono formate da una barra inversa (\) seguita da un carattere. Nella sequenza precedente \n è una sequenza di escare chiamata carattere di nuova riga, che significa "a capo". Le sequenze di escare sono utilizzate per stampare caratteri particolari.

Comando	**Descrizione**
\t	Tabulazione
\b	blank space
\'	Apice singolo (come carattere stampato)
\\	Barra inversa (come carattere stampato)
\n	Avanzamento di riga (a capo)
\?	Punto interrogativo (come carattere stampato)
\"	I doppi apici stampati (come carattere stampato)
\a	Segnale acustico

1.4.3 La funzione system()

La funzione *system()* consente di eseguire comandi della shell del sistema operativo. Per poterla utilizzare all'interno del programma è necessario includere la libreria *stdlib.h*. Il testo letterale che viene passato tra parentesi tonde (argomento della funzione) individua il comando che deve essere eseguito. Ad esempio per attivare il comando "pause" del sistema operativo DOS:

```
system("pause");
```

1.5 Come scrivere un programma

La leggibilità del codice è di fondamentale importanza, anche il sorgente di un programma che esegue pochi compiti potrebbe richiedere la scrittura di diverse migliaia di righe di codice quindi esistono diverse regole o consuetudini che vale la pena rispettare fin dall'inizio con molta rigorosità e pignoleria.

1.5.1 Indentare il programma

Indentare un programma significa mettere in evidenza nel testo del programma mediante opportuni incolonnamenti, i gruppi di istruzioni che svolgono un problema comune, in modo da evidenziare la funzionalità calcolata da ogni gruppo di istruzioni, rispetto al resto del programma.

1.5.2 Commentare il programma

Le frasi prodotte nel corso del progetto per descrivere l'algoritmo a vari livelli di raffinamento possono comparire nella versione definitiva del sorgente sotto forma di **commenti**. Anche se l'algoritmo è stato progettato utilizzando direttamente il linguaggio di programmazione a disposizione è conveniente inserire in ogni caso dei commenti in punti particolari del programma che spieghino cosa il programma fa in quei punti, indipendentemente da come lo fa, cioè dalla particolare strategia scelta.

I commenti non vengono interpretati dal compilatore e quindi non vanno ad incidere sull'operatività o sulla dimensione del file eseguibile finale.

Un commento può essere scritto su una o più righe:

```
/* Commento su una sola riga*/

/* Commento
che occupa più
di una riga*/

printf("Forza Cesena!"); /*Commento che occupa parte di una riga*/
```

Esiste anche una sintassi diversa per commentare i propri programmi utilizzando una doppia barra:

```
// questa riga è un commento

printf("Forza Cesena!"); // il commento inizia con la doppia barra
```

La doppia barra indica che il testo della riga è da considerarsi come un commento. Anche se molti compilatori C supportano questo tipo di sintassi è bene evitarla quando è necessario garantire aperta portabilità.

1.5.3 Usare nomi di variabili autoesplicativi

Per spiegare il ruolo delle variabili, oltre che usare commenti, possiamo scegliere nomi che già da soli ne spieghino il ruolo. Per esempio il nome della variabile *stipendioMensile* è autoesplicativo, la stessa variabile se fosse stata chiamata xMk il programma sarebbe risultato meno comprensibile ma ugualmente funzionante.

1.5.4 Produrre la documentazione nel corso stesso del progetto

E' importantissimo scrivere la documentazione durante lo sviluppo del progetto in quanto documentando le decisioni nello stesso momento in cui vengono prese, si è certi della efficacia e della completezza della documentazione. E' storia di ogni giorno che la documentazione venga prodotta solo al termine del progetto frettolosamente e senza rispettare alcuno standard.