

Primi passi col linguaggio C

Andrea Marin

Università Ca' Foscari Venezia
Laurea in Informatica
Corso di Programmazione part-time

a.a. 2011/2012

Come introdurre un linguaggio di programmazione?

Obiettivi:

- ▶ Introduciamo una macchina astratta in grado di interpretare i programmi scritti nel linguaggio C
- ▶ Descriviamo lo **stato** della macchina astratta e il modo in cui esegue input/output
- ▶ L'esecuzione di un'istruzione altera lo stato della macchina astratta
 - ▶ Descriveremo le istruzioni fondamentali in termini di modifiche dello stato della macchina astratta



Vantaggi:

- ▶ L'uso di una macchina astratta (e della **semantica** come operazioni sul suo stato) consente di definire un linguaggio indipendentemente dalla piattaforma in uso
 - ▶ Per piattaforma intendiamo Macchina fisica + Sistema operativo

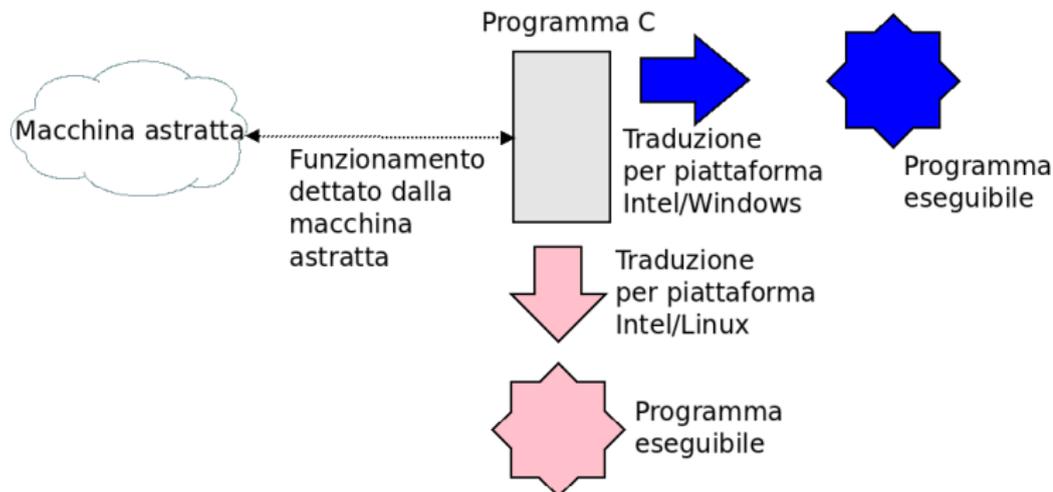
Da risolvere. . .

- ▶ Il programma viene eseguito su una piattaforma specifica!
- ▶ Bisogna tradurre il programma scritto per la macchina astratta in un programma eseguibile sulla piattaforma specifica
- ▶ Questa traduzione prende il nome di **compilazione** del linguaggio



Esempio

Write once, compile everywhere!



La compilazione del programma

- ▶ Il compilatore trasforma le istruzioni del programma in linguaggio macchina
- ▶ Il codice del programma C è detto **codice sorgente**
- ▶ Il codice macchina ottenuto è detto codice oggetto
- ▶ Il codice oggetto è strettamente legato all'hardware

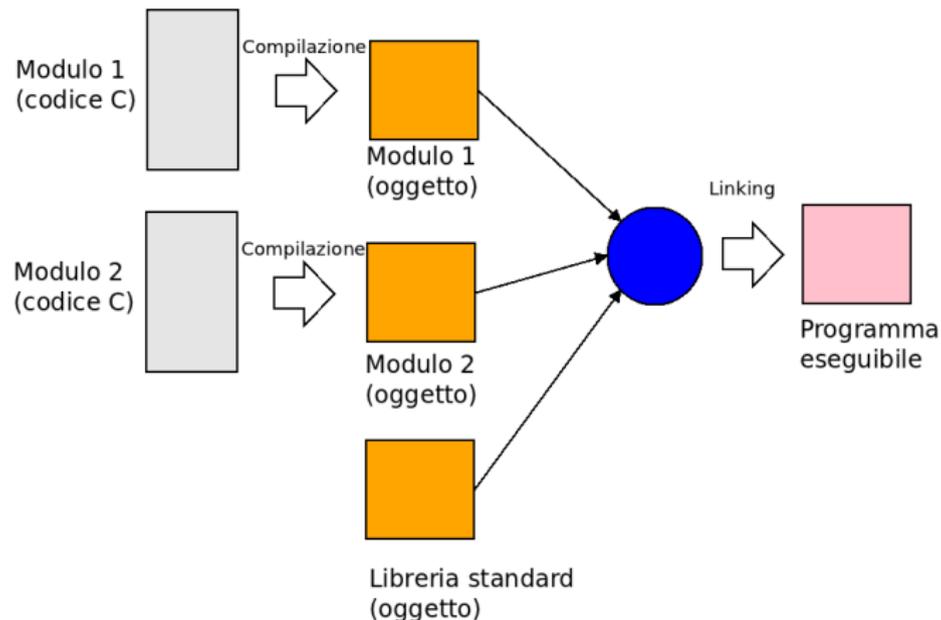


La fase di collegamento (linking)

- ▶ In generale un programma complesso è composto da tante parti che si chiamano **moduli**
- ▶ Ogni modulo può essere compilato separatamente e dà origine ad un **modulo oggetto**
- ▶ Il collegamento tra i moduli viene fatto dal **linker**
- ▶ Questo metodo consente la creazione e riutilizzo di librerie
 - ▶ standard
 - ▶ definite dal programmatore



Schema



Gli header

- ▶ I moduli di un programma possono far riferimento a *funzionalità* di altri moduli
- ▶ A tal fine ogni modulo può dichiarare cosa *sa fare* mediante i file header

Definition (File header)

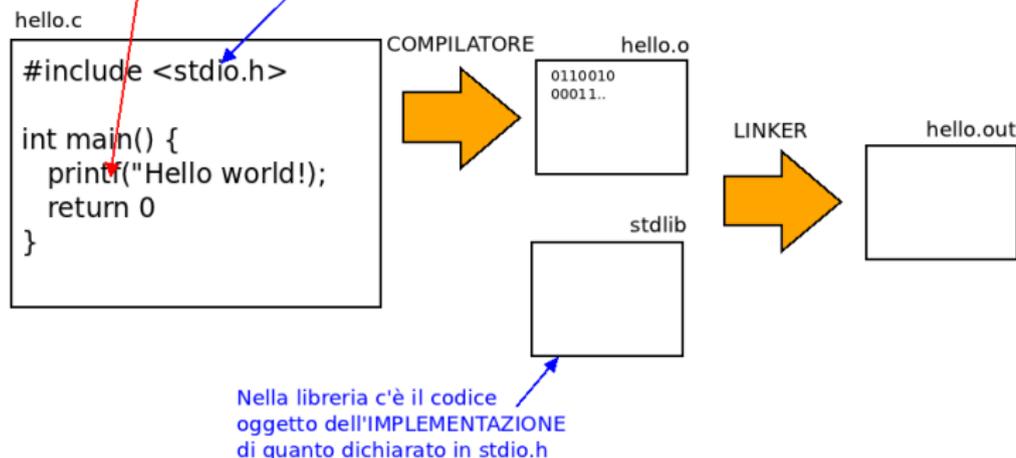
File contenente costanti, direttive di pre-processor, dichiarazioni di strutture, prototipi di funzioni e in alcuni casi l'implementazione dei prototipi delle funzioni.

- ▶ Ciascuno di questi elementi sarà trattato successivamente nel corso

Hello world!

"printf" è definita nella libreria stdio. Il compilatore può ottenere il codice oggetto di hello.c grazie all'header

Chiede di usare le funzionalità definite nella libreria standard stdio e DICHIARATE nel file stdio.h



Le operazioni da console Linux

```
andrea@marand: ~/prova
File Modifica Visualizza Cerca Terminale Aiuto
andrea@marand:~/prova$ ls
hello.c
andrea@marand:~/prova$ cat hello.c
#include <stdio.h>

int main() {
    printf("Hello world\n");
    return 0;
}

andrea@marand:~/prova$ gcc hello.c -ansi -o hello.out
andrea@marand:~/prova$ ls
hello.c hello.out
andrea@marand:~/prova$ ./hello.out
Hello world
andrea@marand:~/prova$
```

- ▶ gcc è il compilatore e linker
- ▶ l'opzione `-ansi` serve ad indicare lo standard di riferimento
- ▶ gcc può anche solo compilare o solo eseguire il linking. In questo caso svolge i due passaggi con un solo comando



Identificatori

- ▶ L'identificatore è un nome attraverso il quale possiamo accedere ad una variabile senza conoscere l'indirizzo della locazione di memoria
- ▶ In una prima fase del corso accederemo alle locazioni di memoria solamente mediante un identificatore
- ▶ Gli identificatori in C cominciano con una lettera e possono contenere lettere, cifre e il carattere _
 - ▶ Sono identificatori: pippo, prima_prova, prova23



Variabile

Definition (Variabile in C)

Una variabile è una cella di memoria caratterizzata da: nome (identificatore), locazione, tipo e valore. In C il valore rappresenta il contenuto informativo della variabile ed è l'unico attributo che può cambiare durante l'esecuzione del programma.

- ▶ Il **tipo** della variabile serve a:
 - ▶ Definire l'insieme dei valori che possono essere memorizzati (interi, reali, testi. . .)
 - ▶ Definire il modo in cui si codifica il dato da memorizzare (incluso il numero di locazioni elementari contigue richieste)
 - ▶ Le operazioni che possono essere svolte su quella variabile



Tipi elementari in C

Nome	esempio valore	dimensione (bytes)	Range
char	'b'	1	-128 a 127
short int	2	-234	-32768 a 32767
int	2 o 4	765	dipende
long int	4 o 8	828277	dipende
float	4	343.23E-2	$1 \cdot 10^{-37}$ to $1 \cdot 10^{37}$ +/-
double	8	-5252.0E-56	$1 \cdot 10^{-308}$ to $1 \cdot 10^{308}$ +/-

Nei tipi `shortint`, `char`, `int`, `long int` si può far precedere la parola chiave `unsigned` per ottenere la rappresentazione senza segno



Codifica dell'informazione (cenni)

- ▶ A seconda del tipo l'informazione viene codificata in sequenze di 0 e 1
- ▶ Per i tipi interi si usa la trasformazione in binario (eventualmente con il complemento a 2)
- ▶ Per la rappresentazione dei numeri in virgola mobile si usa la codifica standard IEEE
- ▶ La rappresentazione dei caratteri avviene mediante la loro posizione nella tabella ASCII
 - ▶ Il codice per la lettera 'A' è il 65
 - ▶ Il codice per la lettera 'a' è il 97
 - ▶ Il codice per la cifra '0' è il 48



Creare una variabile di un certo tipo

- ▶ Per dichiarare una variabile è sufficiente specificarne il tipo seguito dall'identificatore scelto
- ▶ La variabile verrà automaticamente allocata in una porzione di memoria libera
 - ▶ il numero di bytes necessario è ottenuto grazie al tipo
 - ▶ l'identificatore ci consentirà di scrivere e leggere il valore della variabile
 - ▶ Esempio: `int prima_variabile`; dichiara una variabile intera
 - ▶ In una prima fase dichiareremo tutte le variabili prima dell'esecuzione della prima istruzione del programma



Esempio

Il seguente programma dichiara tre variabili che rimangono inutilizzate:

```
1. #include <stdio.h>
2. double a;
3. int prova, inutile;
4. int main(){
5.     printf("Hello world \n ");
6.     return 0;
7. }
```



Acquisire dallo stream di input il valore per una variabile: scanf

- ▶ L'operazione è svolta dalla funzione `scanf` della libreria `stdio`
- ▶ Supponiamo di voler leggere il valore per la variabile `prova` dell'esempio precedente:

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

Nome della variabile in cui inserire il
valore prelevato dallo standard input
preceduto dal simbolo &

%d indica alla funzione `scanf`
di prelevare dallo stream in ingresso
dei caratteri da interpretare con un
intero



Alcuni caratteri di formattazione dell'input/output

Codice	formattazione
c	carattere
d oppure i	intero con segno
f	virgola mobile
s	sequenza di caratteri
u	intero senza segno
p	indirizzo di memoria

- consultare la guida o il manuale per maggiori dettagli!



Scrivere sullo stream di output

- ▶ L'operazione è svolta dalla funzione `printf` della libreria `stdio`
- ▶ Supponiamo di voler stampare il valore della variabile di tipo `float` a dell'esempio precedente

```
printf("%f", a);
```

`%f` indica alla funzione `printf` il modo in cui visualizzare il valore della variabile `a`

Nome della variabile del cui valore si desidera fare l'output



Data di nascita

Scriviamo un programma C che...

- ▶ legga dallo standard input giorno, mese, anno di nascita
- ▶ scriva sullo standard output **Sei nato il gg/mm/aaaa !**
- ▶ utilizziamo tre variabili per memorizzare rispettivamente il giorno, il mese e l'anno



Il programma

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

```
int giorno , mese , anno ;
```

```
int main() {  
    scanf("%d" , &giorno) ;  
    scanf("%d" , &mese) ;  
    scanf("%d" , &anno) ;  
    printf("Se nato il %d/%d/%d \n" , giorno ,  
          mese , anno) ;  
    return 0 ;  
}
```



Esempio interpretazione



prossima istruzione:
`scanf("%d",&giorno)`



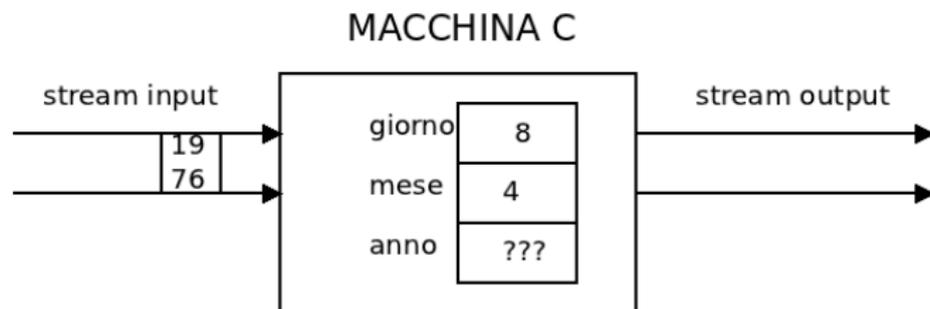
Esempio interpretazione



prossima istruzione:
`scanf("%d",&mese)`



Esempio interpretazione



prossima istruzione:
`scanf("%d",&anno)`

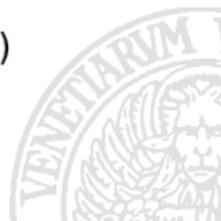


Esempio interpretazione



prossima istruzione:

```
printf("Sei nato il %d/%d/%d \n",giorno,mese,anno)
```



Esempio interpretazione

