

La programmazione strutturata

Stefano Ferrari

Università degli Studi di Milano
stefano.ferrari@unimi.it

Programmazione

anno accademico 2016–2017

Strutturare un programma

```
*  *  
*  *  
****  
*  *  
*  *
```

```
****  
*  
****  
*  
****
```

```
*  
*  
*  
*  
****
```

```
*  
*  
*  
*  
****
```

```
***  
*  *  
*  *  
*  *  
***
```

Scrivere programmi illeggibili è facilissimo

Supponiamo di voler stampare il banner qui a sinistra

Usando le funzioni elementari di stampa (che ancora non abbiamo studiato) si può fare così

```
printf("*");  
printf("_");  
printf("_");  
printf("_");  
printf("*");  
printf("\n");  
printf("*");  
printf("_");  
  
:  
:  
:  
:  
printf("*");  
printf("\n");  
printf("_");  
printf("*");  
printf("*");  
printf("*");  
printf("_");  
printf("\n");
```

5 lettere

5 righe

6 caratteri

4 righe vuote

154 righe di codice!

Strutturare un programma (2)

```
* *      *
```

```
*       *
```

```
*****
```

```
*
```

```
*       *
```

```
*       *
```

```
*****
```

```
*
```

```
*****
```

```
*
```

```
*****
```

```
*
```

```
*
```

```
*
```

```
*
```

```
*****
```

```
*
```

```
*
```

```
*
```

```
*
```

```
*****
```

```
    ***
```

```
*     *
```

```
*     *
```

```
*     *
```

Scrivere programmi illeggibili è facilissimo

Supponiamo di voler stampare il banner qui a sinistra

Si può fare anche così

```
printf("*uuu*\n*uuu*\n*****\n . . . \n*   *\n*   *\n*   *\n*** \n");
```

Una sola linea di 184 caratteri!

Strutturare un programma (3)

```

*      *
*      *
*****
*      *
*      *
*      *

*****
*
*****
*
*****
*
*
*
*
*****
*
*
*
*
*****
      ***
*      *
*      *
*      *
      ****

```

Scrivere programmi illeggibili è facilissimo

Supponiamo di voler stampare il banner qui a sinistra

E ancora si può fare così

```
printf(" *uuu*\n*uuu*\n*****");
printf("\n*uuu*\n*uuu*\n\n");
printf(" *uuu*\n*uuu*\n*****");
printf("\n*uuu*\n*****\n\n");
printf(" *uuuu*\n*uuuu*\n*uuuu*");
printf("\n*uuuu*\n*****\n\n");
printf(" *uuuu*\n*uuuu*\n*uuuu*");
printf("\n*uuuu*\n*****\n\n");
printf(" *uuuu*\n*uuu*\n*uuu*");
printf("\n*uuu*\n*uuu*\n*uuu*");
```

Compatto, ma del tutto incomprensibile!

Approccio top-down

L'**approccio top-down** progetta un **algoritmo** per un problema

- ▶ **partendo dai requisiti** posti dal problema
- ▶ **scomponendo il problema in sottoproblemi gerarchicamente** (cioè i sottoproblemi in sottosottoproblemi, ecc. . .)
- ▶ **arrestandosi al livello dei compiti elementari**

Compiti elementari sono le **operazioni per cui esiste già del codice**

Questo approccio si riflette direttamente nella scrittura del codice

Approccio top-down (2)

“Scrivere HELLO sul video” si può scomporre in

1. **“Scrivere H sul video”**
2. **“Scrivere E sul video”**
3. **“Scrivere L sul video”**
4. **“Scrivere L sul video”**
5. **“Scrivere O sul video”**

Intuiamo che la scomposizione è corretta perché

- ▶ ogni sottocompito è chiaro e indipendente dagli altri
- ▶ alcuni sottocompiti si ripetono (le due scritture di L)

Lo stesso avviene ai livelli inferiori

Approccio top-down (3)

“Scrivere HELLO sul video” si può scomporre in

1. “Scrivere H sul video” si può scomporre in
 - a) “Scrivere * * sul video”
 - b) “Scrivere * * sul video”
 - c) “Scrivere ***** sul video”
 - d) “Scrivere * * sul video”
 - e) “Scrivere * * sul video”
2. “Scrivere E sul video” si può scomporre in
 - a) “Scrivere ***** sul video”
 - b) ...

Approccio top-down (4)

A sua volta, “Scrivere * * sul video” si può scomporre in

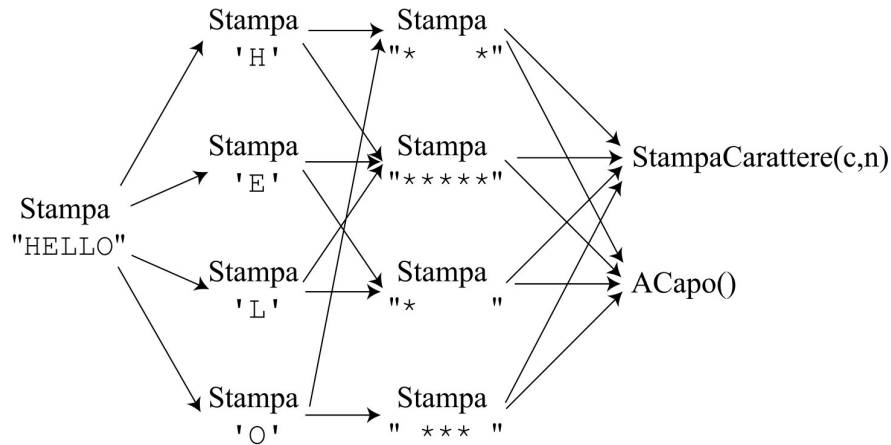
- ▶ “Scrivere * sul video”
- ▶ “Scrivere tre spazi bianchi sul video”
- ▶ “Scrivere * sul video”

La scrittura di un singolo carattere è ovviamente un compito elementare

Noi però possediamo una libreria con una procedura per scrivere sequenze di caratteri uguali: possiamo arrestare prima la scomposizione

L'approccio bottom-up consiste nel dotarsi di procedure che consentono di arrestare la scomposizione in anticipo

Struttura modulare



Struttura modulare (2)

La struttura modulare del codice riflette il progetto top-down

Il codice viene **strutturato in modo da essere gestibile**

1. **si scompone il codice in moduli o blocchi** strutturati gerarchicamente (esattamente come il problema è scomposto in sottoproblemi)
2. **si rendono i moduli comprensibili** adottando **convenzioni**
 - ▶ dichiarazioni che imitano il linguaggio umano
 - ▶ nomi autoesplicativi per funzioni, macro e variabili
 - ▶ corrispondenza biunivoca fra variabili e oggetti
 - ▶ usare spazi, a capi e indentazioni per chiarire il senso
 - ▶ commenti (ultima risorsa)
3. **si rendono i moduli controllabili**
 - ▶ definendoli in modo che abbiano **poche interazioni fra loro**
 - ▶ **esplicitando i requisiti di ogni modulo** (dati e risultati)

Struttura modulare (3)

Si vuole ridurre la comprensione di un programma a

- ▶ comprensione di ciascun modulo
- ▶ comprensione dei rapporti fra moduli

Non esistono regole meccaniche: la semplicità è una conquista

Esercizio

1. Realizzare un programma modulare per la stampa del banner "Hello"
2. Realizzare le funzioni per le altre lettere o simboli
3. Strutturare il programma in modo da poter facilmente includere un font diverso
4. Visitare <http://www.figlet.org/> per trarre spunto per altre estensioni

Proprietà desiderabili

Imporre una struttura significa imporsi consapevolmente delle limitazioni

Ogni modulo dovrebbe godere di queste proprietà:

- ▶ **dimensione**: occupare **meno di una schermata**, così da essere leggibile interamente a colpo d'occhio
- ▶ **profondità**: contenere **al massimo tre livelli annidati di sottoblocchi**, possibilmente meno
- ▶ **coerenza temporale**: contenere **attività concettualmente consecutive**
- ▶ **base condizionale**: contenere **attività richieste nelle stesse condizioni**
- ▶ **condivisione dei dati**: lavorare su un **insieme coerente di dati**
- ▶ **coesione funzionale**: puntare a **un solo scopo specifico**
- ▶ avere **un solo punto di ingresso** e **un solo punto di uscita**

Attività di processo e di gestione

Ogni programma contiene

- ▶ **attività di processo**, che **manipolano i dati fino a ottenere i risultati**
- ▶ **attività di gestione**, che svolgono funzioni accessorie:
 - ▶ **prendono decisioni**: decidono quali attività di processo eseguire in base ai dati o ai risultati parziali
 - ▶ **regolano i passaggi da un modulo all'altro**
 - ▶ **gestiscono risorse** (per es., allocano e deallocano memoria)

Le attività di gestione sono descritte dai **costrutti di controllo**

Costrutti di controllo

La programmazione strutturata consente solo tre costrutti di controllo

1. **esecuzione seriale**: si esegue in un ordine prefissato un blocco di attività di processo
2. **selezione**: si esegue uno tra più blocchi alternativi di attività di processo, scegliendolo in base a una condizione logica sui dati o sui risultati parziali
3. **iterazione**: si esegue ripetutamente un blocco di attività di processo, finché vale una condizione logica sui dati o sui risultati parziali

Dijkstra (1969): **Questi tre costrutti**, combinati opportunamente, **sono sufficienti a realizzare qualsiasi trasformazione da dati a risultati**

(purché calcolabile. . .)