Il Pensiero Computazionale: dai Concetti Generali alla Programmazione Software

Prof. Marco Bernardo

Università di Urbino
Dipartimento di Scienze Pure e Applicate
Sezione di Scienze e Tecnologie dell'Informazione
Corso di Laurea in Informatica Applicata

Cos'è il Pensiero Computazionale?

- Locuzione che caratterizza in maniera sintetica il *contributo culturale* apportato dall'informatica alla società contemporanea.
- Introdotto dalla docente universitaria statunitense Jeannette Wing nel 2006 nell'articolo "Computational Thinking" pubblicato sulle Communications of the ACM (volume 49, numero 3, pagine 33–35).
- Attività intellettuale attraverso cui si definiscono delle procedure che vengono poi attuate da un esecutore, che opera nell'ambito di un contesto specifico per raggiungere degli obiettivi prefissati.
- Il pensiero computazionale è un processo mentale finalizzato alla risoluzione di problemi, costituito dalla combinazione di:
  - metodi caratteristici (analisi dei problemi, rappresentazione dei dati, automazione delle soluzioni)
  - capacità intellettuali (affrontare la complessità, gestire le ambiguità, confrontare le alternative)

  che hanno un valore generale (cioè i benefici si estendono ad altre discipline).

Informatica

- Non solo tecnologie per il trattamento automatico delle informazioni.
- È la scienza che per i problemi di natura computazionale studia:
  - la decidibilità, cioè la risolvibilità in tempo finito per ogni loro istanza dei dati di ingresso;
  - le possibili soluzioni algoritmiche, in caso di decidibilità;
  - i relativi costi in termini delle risorse computazionali fondamentali quali tempo, spazio e comunicazione.
- Esistono davvero dei problemi indecidibili? Sì (Gödel, Turing, Church, ...).
- Aree dell'informatica intesa come scienza:
  - teoria della calcolabilità;
  - teoria della complessità;
  - algoritmica;
  - teoria dei linguaggi formali;
  - teoria degli automi;
  - ...

Computer

- Insieme di dispositivi elettromeccanici programmabili per l'immissione, la memorizzazione, l'elaborazione e l'emissione di informazioni.
- Cambiando il programma, cambia la funzione svolta.
- Veicoli, elettrodomestici, ... svolgono un numero *limitato* di funzioni.
- Hardware: processore, memorie, periferiche di input/output.
- Software: sistema operativo, applicazioni.
- Babbage, 1834: Analytical Engine (computer meccanico a schede perforate).
  Turing, 1936: macchina di Turing universale (modello teorico).
  ENIAC, 1946: stanza di 9×15 metri, 30 tonnellate, interruttori e cavi.
  Von Neumann, 1952: computer a programma memorizzato.
- Evoluzione tecnologica: tubi a vuoto $\rightarrow$ transistor (Elea 9003 Olivetti, CEP) $\rightarrow$ circuiti integrati $\rightarrow$ personal computer tipo desktop (P101 Olivetti) $\rightarrow$ personal computer tipo laptop $\rightarrow$ tablet $\rightarrow$ smartphone $\rightarrow$ ...

Algoritmo

- Sequenza finita di passi interpretabili da un esecutore.
- Descrizione dell'algoritmo vs. esecuzione dell'algoritmo.
- Sebbene il numero di passi elencato sia finito, l'esecuzione di tutti quei passi potrebbe richiedere un tempo non necessariamente finito.
- Nessuna specifica sulla natura dell'esecutore.
- Sebbene di norma si pensi che l'esecutore sia un computer, l'esecutore potrebbe anche essere un sistema biologico.
- Esempi: cuoco che prepara una pietanza seguendo una ricetta, ragazzo che costruisce un modellino leggendo delle istruzioni, ...
- Il termine algoritmo deriva dal nome del matematico persiano Muhammad al-Khwarizmi (IX secolo), autore dell'antico trattato di al-Jabr in cui vengono descritte procedure per risolvere equazioni.
- Concetto formalizzato per la prima volta tramite macchina di Turing.

Programma

- Un algoritmo viene di solito scritto in un misto tra linguaggio naturale e notazione matematica, quindi *non* è interpretabile da un computer.
- Per renderlo interpretabile, bisogna in primo luogo trascriverlo in un linguaggio di programmazione, ottenendo così il programma sorgente.
- Vocabolario e sintassi di un linguaggio di programmazione sono molto più limitati rispetto ad un linguaggio naturale (italiano, inglese, ...).
- In secondo luogo, il programma sorgente deve essere tradotto in un programma eseguibile equivalente, cioè una sequenza di 0 e 1 direttamente interpretabili dal computer.
- Questa operazione viene effettuata dal compilatore del linguaggio.
- Il programma eseguibile viene generato solo se il compilatore *non* rileva errori lessicali, sintattici o semantici nel programma sorgente.
- Ogni programma lavora su dei dati attraverso le proprie istruzioni.

Dati

- Ogni dato è caratterizzato da tre elementi.
- Un identificatore, che individua univocamente il dato all'interno del programma sorgente.
- Un tipo, che stabilisce l'insieme dei valori che il dato può assumere e quindi l'insieme delle operazioni in cui il dato può essere coinvolto nelle istruzioni del programma.
- Un valore, che risulta essere attribuito al dato durante un certo periodo dell'esecuzione del programma e che può eventualmente cambiare durante l'esecuzione stessa.
- I primi due elementi sono sempre specificati nella dichiarazione del dato, il terzo solo in caso di contestuale inizializzazione:

  ◁*tipo*▷ ◁*identificatore*▷;

  ◁*tipo*▷ ◁*identificatore*▷ = ◁*valore iniziale*▷;

Dati Costanti / Variabili

- Ogni dato può essere classificato in due diversi modi a seconda della forma della sua dichiarazione.
- Una costante, quando il valore del dato viene assegnato una volta per tutte in fase di dichiarazione e non può cambiare durante l'esecuzione:

```
const int ALTEZZA_MARCO = 182;
#define   ALTEZZA_MARCO   182
```

- Ogni occorrenza dell'identificatore della costante nelle istruzioni del programma può essere sostituita dal valore della costante stessa.
- Una variabile, quando il valore del dato può essere modificato in ogni momento dell'esecuzione del programma:

```
int altezza;
```

- Una variabile funge da contenitore di un valore mutevole ed è perciò ad essa associata dal compilatore una locazione di memoria.

Dati Semplici / Strutturati

- Ogni dato può essere classificato in due diversi modi a seconda della forma del suo valore.
- Dato semplice, quando il suo valore non è decomponibile:
  - tipo `int` per i numeri interi;
  - tipo `double` per i numeri reali;
  - tipo `boolean` per i valori logici;
  - tipo `char` per i caratteri.
- Dato strutturato, quando il suo valore è composto da più sottovalori:
  - array o string quando i sottovalori sono tutti dello stesso tipo:

```
int altezze[25];
```

  - record quando i sottovalori possono essere di tipi diversi:

```
struct
{
  char nome[10];
  int lune;
  double tempo_rotazione, tempo_orbita;
} pianeta;
```

Istruzioni

- Un programma sorgente è composto da una sequenza di istruzioni che vengono eseguite, una alla volta, nell'ordine in cui sono scritte.
- Ad ogni istruzione di un linguaggio di programmazione corrisponde una sequenza di comandi direttamente interpretabili dal computer, nei quali l'istruzione stessa viene tradotta dal compilatore.
- Le istruzioni acquisiscono dati, li rielaborano e producono risultati.
- Nel paradigma di programmazione imperativo (procedurale o ad oggetti), le istruzioni si dividono in:
  - istruzioni di assegnamento;
  - istruzioni di selezione;
  - istruzioni di ripetizione.
- Le istruzioni finalizzate a svolgere un certo compito possono essere raccolte e organizzate una volta per tutte in un sottoprogramma che viene poi richiamato tutte le volte che serve.

Istruzioni di Assegnamento

- Sintassi:

  ⟨identificatore di variabile⟩ = *⟨espressione⟩*;

- Semantica:
  procedendo da destra a sinistra, per prima cosa viene valutata l'espressione, poi il suo valore viene memorizzato nella variabile, la quale perde quindi il valore che conteneva in precedenza.

- Esempio:

  ```
  x = (y + 2 * z) / 0.5;
  ```

  Se `y` vale `5` e `z` vale `3` subito prima dell'esecuzione dell'istruzione, `x` varrà `22` al termine dell'esecuzione dell'istruzione.

- Nelle espressioni si possono usare tutti gli operatori dell'aritmetica, ai quali si applicano le consuete regole di precedenza e associatività.
- Si possono inoltre utilizzare tante altre funzioni matematiche come elevamento a potenza, radice, logaritmo, ...

Istruzioni di Selezione

- Sintassi:

```
if (◁espressione logica▷)
  ◁istruzione 1▷
else
  ◁istruzione 2▷
```

- Semantica:
  per prima cosa viene valutata l'espressione; se il suo valore è vero, allora viene eseguita la prima istruzione, altrimenti viene eseguita la seconda istruzione.
- Esempio:

```
if (y >= 0)
  x = x + y;
else
  x = x - y;
```

Ad `x` viene in ogni caso aggiunto il valore assoluto di `y`.

Istruzioni di Ripetizione `while`

- Sintassi:

```
while (⟨espressione logica⟩)
  ⟨istruzione⟩
```

- Semantica:
  per prima cosa viene valutata l'espressione, poi viene eseguita l'istruzione fintantoché l'espressione si mantiene vera.
- L'espressione logica funge da *condizione di continuazione*; se essa è inizialmente falsa l'istruzione non viene eseguita nemmeno una volta.
- Sintassi alternativa:

```
do
  ⟨istruzione⟩
while (⟨espressione logica⟩);
```

- Semantica per traduzione:

```
⟨istruzione⟩
while (⟨espressione logica⟩)
  ⟨istruzione⟩
```

Istruzioni di Ripetizione `for`

- Sintassi:

```
for (⟨espressione di inizializzazione⟩;
     ⟨espressione logica⟩;
     ⟨espressione di aggiornamento⟩)
  ⟨istruzione⟩
```

- Semantica per traduzione:

```
⟨espressione di inizializzazione⟩;
while (⟨espressione logica⟩)
{
  ⟨istruzione⟩
  ⟨espressione di aggiornamento⟩;
}
```

- Sia l'espressione di inizializzazione che l'espressione di aggiornamento riguardano le variabili su cui si basa la condizione di continuazione.

Sottoprogrammi

- Vengono chiamati *subroutine*, *procedure*, *funzioni* o *metodi* a seconda dello specifico linguaggio di programmazione.
- Insieme di istruzioni finalizzate a svolgere un certo compito:
  - definite una volta per tutte all'interno del programma sorgente tramite parametri formali (variabili di interfaccia);
  - richiamate ogni volta che serve su parametri effettivi (espressioni) che possono essere diversi da invocazione ad invocazione e vengono sostituiti ai corrispondenti parametri formali.
- Definizione di una funzione:

```
⟨tipo risultato⟩ ⟨id. funzione⟩ (⟨dich. param. formali⟩)
{
  ⟨dichiarazione dati locali⟩
  ⟨istruzioni della funzione⟩
}
```

- Invocazione di una funzione (come istruzione singola o dentro un'espressione):

```
⟨id. funzione⟩ (⟨param. effettivi⟩)
```

Esempio di Sottoprogramma

- Calcolare l'altezza media in centimetri degli studenti di una classe:

```
int calcola_altezza_media(int altezze[],
                          int numero_studenti)
{
  int somma_altezze = 0,
      altezza_media,
      i;

  for (i = 0; i < numero_studenti; i = i + 1)
    somma_altezze = somma_altezze + altezze[i];
  altezza_media = somma_altezze / numero_studenti;
  return(altezza_media);
}
```

Variante dell'Esempio

- Calcolare l'altezza media in centimetri degli studenti di una classe *scartando le altezze che si trovano al di fuori di un certo intervallo*:

```
int calcola_altezza_media(int altezze[],
                          int numero_studenti)
{
  int somma_altezze = 0,
      numero_scarti = 0,
      altezza_media,
      i;

  for (i = 0; i < numero_studenti; i = i + 1)
    if (altezze[i] >= 150 && altezze[i] <= 180)
      somma_altezze = somma_altezze + altezze[i];
    else
      numero_scarti = numero_scarti + 1;
  numero_studenti = numero_studenti - numero_scarti;
  altezza_media = somma_altezze / numero_studenti;
  return(altezza_media);
}
```

Conclusioni

- Non siate solo fruitori passivi delle tecnologie informatiche!
- Liberate il vostro pensiero computazionale!
- Divertitevi a sviluppare soluzioni algoritmiche ai problemi!