Fondamenti di programmazione

Di: Edoardo Lo Iacono

Contents

1 I programmi

I programmi informatici sono oggi materia comune per tutti quanti, sono programmi le applicazioni che usiamo sul nostro cellulare come i siti web su cui navighiamo. Compresa l'importanza di essi dobbiamo capire bene cosa significhi scrivere un programma per poter proseguire ed andare ad analizzare le diverse parti che lo compongono.

Possiamo definire un programma come un insieme di istruzioni tradotte in un determinato linguaggio di programmazione. Queste istruzioni sequenziali vengono comunemente chiamate **algoritmo**.

Queste istruzioni prendono solitamente in considerazione dei dati in ingresso, chiamati **input** per restituirli sotto forma di dati in uscita detti **output**. Per comprendere meglio questo concetto andiamo a vedere un paio di esempi.

- Un primo esempio slegato dal mondo dell'informatica ma molto attinente è quello di una ricetta, in questo caso i nostri dati in input saranno gli ingredienti presi separatamente, il nostro algoritmo sarà la ricetta vera e propria, ovvero le istruzioni da seguire e infine il piatto portato a termine sarà il nostro output.
- Dato un semplice problema che ci richiede di calcolare la media aritmetica di 3 numeri in ingresso, i nostri dati in input saranno i 3 numeri in quanto vengono inseriti dall'utente, l'algoritmo sarà l'insieme di operazioni che portano a calcolare la media, ovvero la somma dei 3 valori in ingresso divisa per 3 ($\frac{v1+v2+v3}{3}$), l'output sarà la media vera e propria.

Visti questi esempi possiamo notare come un'algoritmo sia quindi un insieme finito di azioni che risolvono un determinato problema (cucinare,calcolare la media) andando a trasformare dei dati in input (ingredienti, valori) in dati in output (piatto, media).

2 Le variabili

Il modo più semplice per pensare ad una variabile è figurarsela come un **contenitore** allocato (ovvero memorizzato) nella memoria dell'elaboratore. Le variabili possono essere di due tipi in base al linguaggio di programmazione:

- **Tipizzate** = Dire che una variabile è tipizzata significa specificare che dentro il nostro contenitore sarà possibile inserire *solo* elementi di un determinato tipo, parleremo tra poco dei diversi tipi di variabile. Il tipo di elemento scelto viene in questi casi specificato in fase di creazione di una variabile.
- **Non tipizzate** = Al contrario delle variabili tipizzate non è necessario stabilire da subito quale tipo di elementi sarà possibile mettere all'interno della nostra variabile.

Per comprendere meglio il concetto teorico di variabile riprendiamo l'esempio della media fatto prima; in questo caso le nostre variabili saranno i 3 valori che diamo in ingresso le quali avranno al loro interno un valore di tipo numerico.

A ogni variabile è associato un nome detto **identificatore** che identifica in maniera univoca la variabile nel programma e, come detto un tipo il quale indica la natura degli elementi interni a questi contenitori. Sono chiamate variabili in quanto al loro interno sono presenti valori che possono cambiare durante l'esecuzione del programma, analoghe alle variabili ma con valori che rimangono invece statici troviamo le *costanti*.

La sintassi per dichiarare una variabile, comune alla maggior parte dei linguaggi di programmazione tipizzati è la seguente:

```
Tipo nomeVariabile = eventualeValore;
```

Il valore interno della variabile può essere definito staticamente mettendolo dopo l'uguale oppure dinamicamente nel codice, in questo caso la dichiarazione sarà soltanto Tipo nomeVariabile;

2.1 Standard di notazione delle variabili

Uno dei problemi più comuni per chi è alle prime armi nella programmazione di qualsiasi tipo è quello dovuto a delle incongruenze nei nomi delle variabili; prima di chiarire meglio quello che intendo specifichiamo una cosa: i nomi delle variabili sono del tutto **arbitrari**, questo significa che è a discrezione del programmatore l'identificatore che viene dato alla variabile. Proprio per questo è una buona norma usare dei nomi **parlanti**. Si definisce parlante un nome che renda facile comprendere lo scopo ed il contenuto di una variabile anche in assenza del resto del codice. Tornando al discorso iniziale, uno dei principali errori di programmatori principianti e non è quello di usare caratteri che non sono consentiti o possono creare problemi nel nome di una variabile, è importante tenere a mente che non vanno mai usati *spazi* ed *accenti*.

Per ovviare all'assenza di spazi, gli sviluppatori hanno trovato altri modi di scrivere i nomi delle variabili (e di altre entità che vedremo in seguito) nel codice, tra i più comuni troviamo:

Stile	**Descrizione**	**Esempio**
Camel Case	Inizio minuscolo, parole successive attaccate con iniziale maiuscola.	$ciaoMondo$
Snake Case	Parole minuscole separate da trattino basso.	$ciao\_mondo$
Kebab Case	Parole minuscole separate da trattino normale.	$ciao - mondo$
Pascal Case	Simile al Camel Case, ma con la prima lettera maiuscola.	$CiaoMondo$

2.2 Principali tipi di variabili

I principali tipi di variabili, comuni a pressoché tutti i linguaggi di programmazione sono:

- **int**: Rappresenta i numeri interi (Integer)
- **double**: Rappresenta i numeri reali con virgola mobile
- **char**: Rappresenta un singolo carattere, si scrive tra apici singoli (')
- **string**: Rappresenta una stringa ovvero un insieme (o per meglio dire un array, concetto che vedremo meglio in seguito) di caratteri, si scrive tra doppi apici ("")
- **bool**: Rappresenta un valore binario che può essere 'true' oppure 'false'

3 Le condizioni

Per dare una definizione balisare abbiamo detto che un algoritmo è un insieme di operazioni eseguite in seguenza l'una dopo l'altra, questa definizione non è però del tutto vera, o per meglio dire non lo è sempre. Con l'entrata in gioco delle condizioni abbiamo parti di codice che vengono o meno processati in base alla validità o meno di una condizione. E' quindi normale trovare parti di codice che in alcune esecuzioni possono venire completamente ignorate.

Prima di approfondire il discorso sui comandi condizionali credo che sia necessario fare un breve ripasso di **logica**, in quanto ci tornerà molto utile in seguito: Tra i principali operatori logici troviamo la **AND**, espressa solitamente con il simbolo $\wedge$ e la **OR**, che possiamo dividere in una disgiunzione inclusiva (in latino *vel*), rappresentata con il simbolo $\vee$ e la disgiunzione esclusiva (in latino *aut* la quale accetta che solamente uno dei due valori sia vero e non entrambi). Poniamo di seguito le tavole di verità di questi due operatori:

AND

A	B	A∧B
V	F	F
V	V	V
F	V	F
F	F	F

OR

A	B	A∨B
V	F	V
V	V	V
F	V	V
F	F	F

Fatto questo brevissimo richiamo alla logica, torniamo a parlare delle operazioni condizionali, nella maggior parte dei linguaggi di programmazione, il blocco condizionale si sviluppa nel seguente modo

```
if(condizione)
{
    ... //Codice eseguito SOLAMENTE se la condizione tra parentesi
        risulta vera
}
else
{
    ... //Codice eseguito SOLAMENTE se la condizione tra parentesi
        risulta falsa
}
//Codice eseguito sempre
```

Possiamo avere però anche dei casi più particolari in cui la nostra condizione si ramifica in più condizioni, pensiamo per esempio di voler applicare uno sconto se il numero di biglietti comprati è superiore a 5 ed uno sconto ancora maggiore se è superiore a 10, non ci basterà qui una sola condizione. In questo caso possiamo usare una struttura del seguente tipo:

```
if(nPartecipanti >= 10)
{
    //Applico lo sconto del 50\%
}
else if(nPartecipanti >= 5)
{
    //La condizione risulta falsa rispetto alla prima if e quindi fa
        parte dell'else, ma poniamo qui una nuova condizione
}
```

Nel caso in cui si venga a creare una condizione per cui sono necessarie molte 'else if', conviene usare il blocco **switch**, che subisce leggere variazioni in ogni linguaggio di programmazione ma permette, in base al valore di una variabile di eseguire operazioni diverse tra di loro.