

Guida alla programmazione in Ruby
Parte Prima

Tratta da <http://stelk.altervista.org/>

Indice

- Cos'e' la programmazione?*
- Tipi di linguaggi*
- Vantaggi e svantaggi dei vari tipi di linguaggi*
- Il nostro primo programma, "Hello World" in Ruby*
- Il paradigma Object Oriented*
- Le variabili*
- L'input dei dati in Ruby*
- Risorse in rete*

Cos'è la programmazione?

La programmazione è quell'insieme di attività che porta alla creazione di un software, usando un determinato *linguaggio di programmazione*.

Un linguaggio di programmazione, come tutti i linguaggi, segue delle regole di sintassi e di semantica, ed è utilizzato per creare programmi e soprattutto per facilitare la scrittura di software ai programmatori.

Infatti, i computer al loro interno funzionano manipolando solo dati in formato binario cioè costituiti solo da 0 e 1; poiché la manipolazione di questo tipo di dati e la codifica di algoritmi, usando il sistema binario, sono molto difficili per noi, sono stati creati appositi linguaggi di programmazione che servono da tramite tra il computer e l'uomo.

Tipi di linguaggi

Esistono vari linguaggi e vari tipi di linguaggi. Si possono dividere i linguaggi in due tipi principali: *interpretati* e *compilati*.

I linguaggi compilati sono quei linguaggi che vengono tradotti in codice macchina (in formato binario) da un apposito programma detto *compilatore*.

Una volta tradotto in codice binario, il programma può essere eseguito solo sul computer sul quale è stato compilato o quantomeno su computer che utilizzano componenti hardware (processori) o software (sistemi operativi) simili.

I programmi scritti nei linguaggi interpretati invece, non necessitano di essere tradotti in linguaggio macchina, poiché vengono eseguiti "al volo" da un programma detto *interprete* che deve essere installato sulla macchina sulla quale si vuole eseguire il programma

Vantaggi e svantaggi dei vari tipi di linguaggi

Vantaggi e svantaggi dei linguaggi compilati.

Uno dei vantaggi principali dei linguaggi compilati è la velocità: un programma scritto in codice binario viene eseguito molto più velocemente di un programma che deve essere "interpretato" prima di essere eseguito.

Un grande svantaggio però è la portabilità, cioè la possibilità di eseguire un programma su piattaforme (sistemi operativi) diverse: un linguaggio compilato infatti può essere eseguito solo su una determinata piattaforma e quindi per eseguire lo stesso programma su una piattaforma diversa bisogna ricompilarlo.

Vantaggi e svantaggi dei linguaggi interpretati.

Ora appaiono evidenti gli svantaggi e i vantaggi dei linguaggi interpretati: la velocità è minore rispetto a quella dei linguaggi compilati, per questo essi vengono usati principalmente in programmi che non necessitano una grande velocità di esecuzione. Il vantaggio più importante è la portabilità: se l'interprete può essere eseguito su più piattaforme anche i programmi scritti in un determinato linguaggio possono essere eseguiti su quelle piattaforme.

Il nostro primo programma, “Hello World” in Ruby

Ruby e' un linguaggio *interpretato* inizialmente sviluppato in Giappone, il suo interprete e' stato scritto in linguaggio C ed e' rilasciato sotto la licenza GPL¹.

Ruby puo' essere eseguito su varie piattaforme, potete trovare la versione per la vostra su:

<http://www.ruby-lang.org/>

Dopo aver scaricato e installato l'interprete possiamo subito partire e creare il nostro primo programma. Aprite il vostro editor di testo preferito o usatene uno incluso nella distribuzione Ruby per iniziare a scrivere.

Il nostro primo programma, come impone la tradizione, sara' un “Hello World” cioe' un programma che ci introduce nel mondo della programmazione semplicemente stampando la scritta “Hello World” sullo schermo.

Ci sono varie istruzioni per stampare stringhe sullo schermo la prima che vedremo e' puts: ecco il suo utilizzo:

```
puts “Hello World”
```

Dopo aver scritto questo nell'editor di testo e averlo salvato, potete farlo eseguire all'interprete dalla linea di comando (ms-dos o una shell linux) digitando:

```
ruby nomefile
```

Dove, ovviamente, nomefile e' il nome del file che contiene il codice che volete eseguire.

Eseguendo il programma che abbiamo scritto prima vedremo che viene stampata sullo schermo la scritta “Hello World”.

La funzione puts prende quindi come argomento una stringa racchiusa tra virgolette e la stampa sullo schermo accordando a essa un ritorno a capo.

Un'altra funzione simile a puts e' print, il suo uso e' praticamente identico solo che essa non accoda il carattere di nuova linea alla fine della stringa, esempio:

```
puts “Hello”           risultato:      Hello
puts “World”          risultato:      World

print “Hello “        risultato:      Hello World
print “World”
```

Come si vede dall'output delle due istruzioni puts aggiunge un ritorno a capo alla fine della stringa mentre print lascia la stringa invariata.

¹ Licenza sotto la quale viene rilasciato il software open source, per maggiori informazioni: <http://www.gnu.org/>

Il paradigma Object Oriented

Un linguaggio di programmazione e' definito un linguaggio Object Oriented, cioe' orientato agli oggetti, quando i dati vengono trattati come oggetti, ognuno con le proprie caratteristiche e con le proprie funzioni.

Per fare un paragone con la nostra vita reale, un oggetto porta, potrebbe avere alcune caratteristiche come il materiale e la grandezza, e alcune funzioni come aprirsi, chiudersi e chiudersi a chiave.

Allo stesso modo, nei linguaggi OO (Object Oriented), ci sono oggetti con determinate caratteristiche e funzioni, un esempio potrebbe essere un oggetti File, che contiene informazioni sul contenuto di un file, sulla grandezza etc... e funzioni per aprirlo, chiuderlo, eliminarlo.

Ruby e' un linguaggio totalmente OO e anche i tipi di dati piu' semplici, come numeri interi o stringhe (sequenze di caratteri) vengono trattati come oggetti.

Questa piccola introduzione servira' semplicemente per capire meglio i contenuti successivi. Ora possiamo alle variabili.

Le variabili

Le variabili in tutti i linguaggi di programmazione, sono dei "contenitori" di memoria che servono a contenere dati inseriti dall'utente o generati dal programma.

Una variabile puo' contenere vari tipi di dati: numeri interi, stringhe o numeri reali.

Per creare una variabile in Ruby basta scrivere:

```
variabile1 = 10
```

oppure:

```
indirizzo = "Via degli Agricoltori, 56"
```

Nel primo caso abbiamo una variabile di nome 'variabile1' che contiene il valore 10.

Nel secondo la variabile di nome 'indirizzo' contiene una stringa, cioe' una sequenza di caratteri. Il nome di una variabile, chiamato identificatore, ci servira' durante tutto il programma per accedere al contenuto della variabile o per apportare ad esso alcune modifiche.

Le altre informazioni sull'uso delle variabili emergeranno durante la guida, poiche' in Ruby tutto viene trattato come un oggetto, servono delle conoscenze della programmazione ad oggetti per capire al meglio alcuni contenuti.

L'input dei dati in Ruby

Abbiamo visto come un programma possa stampare e mandare output sullo schermo, "salvare" dei dati nelle variabili, ora vediamo come l'input dei dati viene gestito in Ruby. Se vogliamo ricevere dell'input da un dispositivo, in questo caso la tastiera, dobbiamo utilizzare delle specifiche funzioni che ci vengono fornite da Ruby. Una di queste e' gets. Ecco un semplice programma che saluta l'utente:

```
print "Come ti chiami? "  
nome = gets  
print "Ciao " + nome
```

La funzione gets blocca il programma fino a che non riceve una linea di input. Alla seconda linea facciamo in modo che l'input inserito dall'utente venga conservato nella variabile 'nome'.

Successivamente stampiamo sullo schermo, "Ciao " seguito dal nome che l'utente ha inserito. Notate che in questo caso l'operatore + non svolge ovviamente un'operazione matematica, ma trattandosi di due stringhe ("Ciao " e il nome inserito) si limita a concatenarle.

Ruby fornisce per l'input molte altre funzioni come read, readline, readlines, etc... per maggiori informazioni e' possibile consultare la documentazione.

Risorse in rete

<http://www.rubycentral.com/>

<http://www.ruby-lang.org/>

<http://ruby-it.org/>

<http://stelk.altervista.org/>