

# Imparare a programmare con Haiku

Scritto da DarkWyrn

Traduzione in italiano, per Haiku Italia, a cura di

Giuseppe Gargaro & Francesca Mora

## Lezione 1

La programmazione prevede di comunicare le vostre idee al computer. L'unico problema è che i computer sono stupidi. Faranno esattamente quello che hai detto e non quello che vuoi dire a meno che ciò che vuoi dire, in realtà, sia proprio quello che hai detto loro. Se non capite completamente questo punto - credetemi - lo capirete presto.

Il problema nella comunicazione con il computer è che non parla una lingua umana. Tutto ciò che sa è cosa fare quando viene fornita una particolare istruzione numerica. Per le persone è molto duro farlo, perciò scriviamo in una lingua accessibile alle persone e il computer la trasforma in qualcosa che può comprendere. La traduzione può avvenire quando il programma viene eseguito, ed è chiamato linguaggio interpretato, o in via preliminare come avviene nei linguaggi compilati. C e C++ sono linguaggi compilati, così quando scrivete programmi C++, scrivete in un linguaggio umano (**codice sorgente**) e il compilatore lo trasforma in istruzioni per il computer (**codice macchina**).

In C++, tutte le istruzioni del computer sono raggruppate in blocchi chiamati **funzioni**. Ecco un esempio:

```
void myFunction(void)
{
}
```

Le funzioni possono o meno, richiedere dati per fare il loro lavoro, e possono o meno, restituire un risultato quando hanno fatto. Questa funzione, chiamata myFunction, non richiede dati e non restituisce nulla. Inoltre non fa nulla, ma ciò è OK. Ecco il formato per la definizione di una funzione.

```
<outdata> <functionname>(indata)
{
    <instructions to do the function>
}
```

I dati di input sono sempre all'interno di una coppia di parentesi e tutte le istruzioni della funzione andranno dentro le parentesi graffe. Per riferimento futuro, tutte le parentesi e le parentesi graffe devono apparire in coppia.

In ogni caso, al compilatore non importa quanto spazio c'è tra tutte queste cose, per cui c'è spazio abbondante per rendere il vostro codice (il)leggibile come volete. Potremmo comprimere tutto questo insieme così:

```
void myFunction(void){}
```

ed avrebbe ancora lo stesso risultato. Perché lo spazio bianco, come si chiama, non è importante; ogni istruzione (da non confondere con una funzione) in C++ è seguita da un punto e virgola.

Per quel che vale, useremo uno stile di scrittura del codice che è molto simile a quello utilizzato dagli sviluppatori di Haiku con qualche ritocco. Per esempio, il codice posto tra una serie di parentesi graffe è sempre rientrato un livello usando il tasto tab.

Ecco la nostra prima funzione che fa davvero qualcosa:

```
int TwoPlusTwo(void)
{
    return 2 + 2;
}
```

Questa funzione, chiamata `TwoPlusTwo`, non ha bisogno di dati in ingresso, ma restituisce un risultato. Il risultato di questa funzione è sempre un numero intero - qualsiasi numero senza un punto decimale. Mentre per le persone un numero è un numero, C++ è molto particolare per quanto riguarda il tener traccia dei tipi di dati passati.

Ogni programma ha una funzione che deve essere definita: `main`. Può essere definita (come dire al computer cosa fare per la funzione `main()`) in un paio di modi diversi, ma useremo il più semplice:

```
int main(void)
{
    return 1;
}
```

Questo programma una volta compilato ed eseguito, non stampa niente; ma restituisce 1 al sistema operativo quando finisce.

Compiliamo questo programma. Salvate il codice sopra in un file codice sorgente chiamato `ReturnOne.cpp`.

Aprire il terminale e spostatevi sulla cartella che lo contiene e scrivete quanto segue:

```
gcc -o ReturnOne ReturnOne.cpp
```

Questo dice al gcc (GNU Compiler Collection) che state per fare un programma chiamato `ReturnOne` (specificato dal `-o ReturnOne`) e avete intenzione di utilizzare `ReturnOne.cpp` per farlo.

Le funzioni possono anche chiamare altre funzioni, ma solo se il computer sa che esistono. Questo genererà un errore durante la compilazione:

```
int main(void)
{
    PushTheRedButton();
    return 1;
}
```

Salvate come `RedButton.cpp` e compilatelo con questo comando:

```
gcc -o RedButton RedButton.cpp
```

Il computer non riconosce la funzione `PushTheRedButton()`, quindi non sa cosa fare quando la vede e si lamenta. Se diciamo al computer cosa fa, sarà compilato correttamente. Cambiate `RedButton.cpp` come segue e compilerà:

