

Dcoder (autore: Vittorio Albertoni)

Premessa

Dcoder è una app per Android che ci consente di scrivere programmi per computer in oltre trenta linguaggi diversi e di sperimentarli su una apparecchiatura (tablet o smartphone) dotata del sistema operativo Android.

Va subito chiarito che i programmi che scriviamo con Dcoder non sono app e non girano su Android ma vengono compilati e fatti funzionare in un ambiente virtuale cui la app si collega via Internet. Per fare funzionare Dcoder è pertanto necessario essere collegati a Internet.

Per occuparci di programmazione per Android, cioè per scrivere programmi (app) che girano sul sistema operativo Android serve altro. In proposito rimando ai miei articoli «Importante riconoscimento per Open JDK» dell'aprile 2016 e relativo allegato «java_android», «Basic su Android» del novembre 2015 e «Python su Android» del giugno 2015 e relativo allegato «sl4a», pubblicati sul mio blog <https://vittal.it> e archiviati nella categoria Programmazione.

Altra limitazione sta nel fatto che con Dcoder non possiamo produrre programmi con interfaccia grafica (GUI) ma dobbiamo lavorare di tastiera e riga di comando, gestendo in modo del tutto particolare l'eventuale interazione di input richiesta dal programma.

Quanto basta, comunque, per avere a disposizione un potente strumento di studio e di pratica di programmazione, preziosissimo per la messa a punto di particolari funzioni e algoritmi che potremo poi inserire in programmi più complessi scritti altrove, magari arricchendoli con GUI. Anche interi programmi di limitata complessità e interattività possono essere scritti e sperimentati con Dcoder.

Il tutto fattibile su uno smartphone, per esempio per ingannare il tempo di un viaggio in treno.

Indice

1	Installazione	1
2	I menu	2
3	L'editor dei programmi	2
4	La compilazione	3
5	Il problema dell'input	4

1 Installazione

L'installazione richiede Android 4.0.3 o superiore e può semplicemente avvenire cercando Dcoder su GooglePlay.

I siti di riferimento sono <http://dcoder.tech/> e <https://paprbit.com/>.

All'indirizzo <https://www.apkmonk.com/app/com.paprbit.dcoder/> troviamo il file .apk della app nel caso volessimo compiere l'installazione manualmente su una apparecchiatura reale o su un simulatore.

Dobbiamo questo intelligente prodotto a Ankush Chugh, fondatore e CEO della Paprbit, ed alla sua équipe.

Non siamo in presenza di software libero e la app gratuita, che ogni tanto ci somministra qualche messaggio pubblicitario e ci invita ad upgradare, ha alcune limitazioni che scompaiono con un acquisto in-app (canone mensile € 0,89, canone annuale € 7,99).

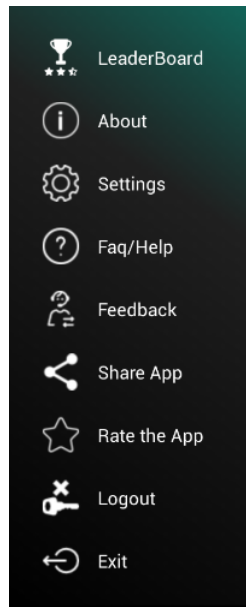
Appena effettuata l'installazione occorre aprire un account indicando l'indirizzo di posta elettronica e scegliendo una password.

Per lavorare, almeno al momento della compilazione, occorre essere collegati a Internet e logati.

2 I menu

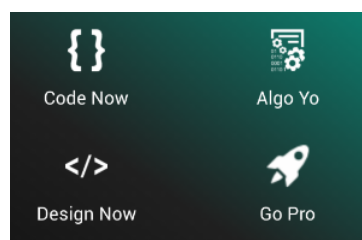
Possiamo suddividere il ricco menu che ci si presenta all'apertura della app in due raggruppamenti.

Il primo riguarda incombenze preparatorie e indicazioni varie e compare in verticale così



Mi pare non vi sia bisogno di commenti particolari in quanto le voci parlano da sé. Dove vediamo scritto LOGOUT, se non fossimo logati come lo ero io al momento della ripresa, vedremmo scritto LOGIN.

Immediatamente sopra questo raggruppamento abbiamo quest'altro



Cliccando sull'icona in basso a destra apriamo la finestra di dialogo per upgradare l'app alla versione pro.

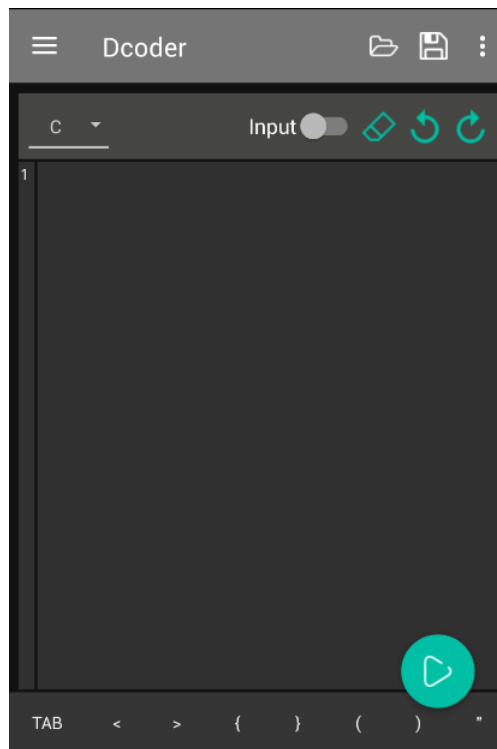
L'icona in alto a destra (ALGO YO) ci introduce nella zona didattica. Vi troviamo enunciati molti problemi tentando di risolvere i quali possiamo sviluppare il nostro pensiero computazionale; essi sono divisi in tre raggruppamenti: facile, medio e difficile.

Per lavorare abbiamo le due icone sulla sinistra: quella in alto (CODE NOW) apre l'editor per scrivere codice in uno dei 32 linguaggi contemplati e quella in basso (DESIGN NOW) apre un editor dedicato specificamente a html, css e JavaScript.

Ciascuno di questi editor è a sua volta dotato di menu che ci fanno accedere a funzioni per l'editing, la compilazione dei programmi, la memorizzazione, ecc.

3 L'editor dei programmi

La finestra dell'editor si presenta così



Nella barra in alto abbiamo, sulla sinistra, un'icona con tre barrette orizzontali, cliccando sulla quale torniamo alla finestra dei menu. Sulla destra abbiamo tre icone, rispettivamente dedicate all'apertura di file precedentemente memorizzati, alla memorizzazione del file in lavorazione e all'apertura di un menu dedicato a funzioni di copia e incolla, chiusura del file. La memorizzazione dei file avviene sulla SDcard (la funzione Salva potrebbe non avere una icona dedicata ma essere accessibile dall'ultimo menu sulla destra).

Nella barra degli strumenti del vero e proprio editor abbiamo un primo menu a sinistra che utilizziamo per scegliere il linguaggio con cui intendiamo codificare. L'illustrazione evidenzia la scelta del linguaggio C e premendo il triangolino sulla destra apriamo l'elenco di tutti i linguaggi disponibili, tra i quali possiamo scegliere.

Effettuata la scelta, nella zona dell'editing compare un piccolo modello di codice teso a rammentare i punti salienti della sintassi del linguaggio scelto (nell'illustrazione non compare in quanto è già stata aperta la zona di lavoro).

Abbiamo poi la scritta INPUT con di fianco un commutatore che attiva o disattiva la scelta; nell'illustrazione la scelta è disattivata.

Segue un'icona cliccando sulla quale apriamo una nuova finestra di lavoro: ci viene offerta la possibilità di caricarvi un modello, ma quando avremo un minimo di conoscenza del linguaggio con cui vogliamo lavorare potremo benissimo farne a meno.

Infine abbiamo due icone che ci consentono di passare da una pagina all'altra dell'editor.

Nella barra inferiore della finestra abbiamo una serie di simboli utili per la scrittura del codice: servono per evitare laboriose ricerche sulla tastiera.

Appena sopra questa barra, sulla destra, abbiamo l'icona in tondo azzurro che dà avvio alla compilazione del codice che abbiamo scritto.

4 La compilazione

La compilazione è la fase in cui è assolutamente necessario avere attivato la connessione a Internet.

Cliccando sull'icona in tondo azzurro che si trova in basso a destra della finestra dell'editor, in questa stessa zona, dopo qualche istante, si apre la finestra dell'output, trascinabile verso l'alto se necessario per leggerne tutto il contenuto.

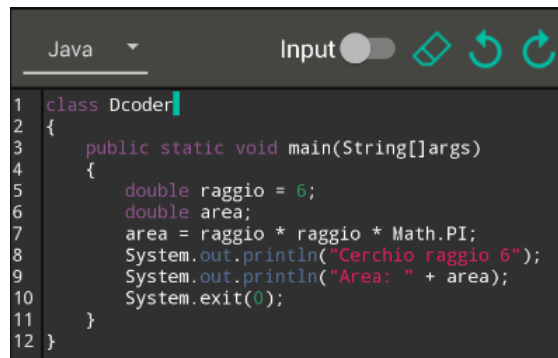
Se il codice è stato scritto senza errori vi troveremo l'output così come lo avevamo programmato.




In caso contrario ci verranno indicati i luoghi e le incongruenze trovate dal compilatore in modo che siamo orientati a capire i tipi di errore che abbiamo commesso nella codifica.

5 Il problema dell'input

Se il codice è costruito in modo che non vi sia bisogno che l'utente introduca dati da elaborare (input), basta scrivere il codice e compilarlo per avere l'output.

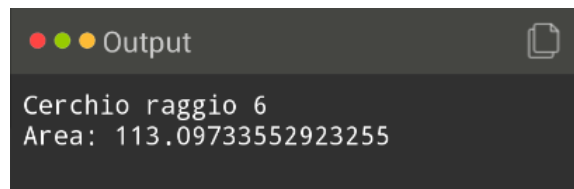
In questo esempio




```
Java Input      
1 class Dcoder  
2 {  
3     public static void main(String[] args)  
4     {  
5         double raggio = 6;  
6         double area;  
7         area = raggio * raggio * Math.PI;  
8         System.out.println("Cerchio raggio 6");  
9         System.out.println("Area: " + area);  
10        System.exit(0);  
11    }  
12 }
```

abbiamo un codice, scritto nel linguaggio Java, per calcolare l'area del cerchio¹. Nel codice stesso è indicato il raggio del cerchio (6). Pertanto il codice calcola l'area di un cerchio di raggio 6 senza bisogno che l'utente inserisca alcun dato.

Con l'input disattivato (vediamo il relativo commutatore con il pallocco grigio sulla sinistra), premiamo l'icona della compilazione e ci troviamo questa finestrella di output



```
Output   
Cerchio raggio 6  
Area: 113.09733552923255
```

dove, evidentemente avendo codificato senza errori, troviamo il risultato dell'elaborazione, evidenziato esattamente come previsto nella codifica.

Ma potremmo essere interessati ad un programma più aperto, per esempio che calcoli sì l'area del cerchio, ma non del cerchio di raggio 6 ma di un cerchio qualsiasi, con raggio indicato di volta in volta dall'utente.

Normalmente questo problema si risolve con una codifica capace di instaurare un dialogo tra il computer e l'utente, in modo che il computer, prima di avviare l'elaborazione, scriva un messaggio chiedendo all'utente di inserire il dato o i dati di cui ha bisogno e l'utente li inserisca².

Questo dialogo non è possibile instaurarlo usando Dcoder, per cui il problema dell'input dobbiamo impostarlo in altro modo.

Faccio un esempio, per semplicità in Python.

Per chiedere all'utente di inserire il raggio del cerchio di cui calcolare l'area e memorizzare il dato nella variabile raggio dovrei scrivere

¹Si osserverà che la classe, nonostante sia destinata ad ospitare un codice per calcolare l'area del cerchio, sia denominata Dcoder anziché, per esempio e come verrebbe spontaneo, Cerchio. Ciò è dovuto ad una esigenza di funzionamento di Dcoder che, nel linguaggio Java, impone che la classe contenente il metodo main sia chiamata Dcoder.

²E' questa la struttura dialogale tipica dei programmi a riga di comando. Nei programmi con GUI, comunque, poco cambia: l'utente inserisce i dati in una finestrella di testo anziché nella riga del terminale.

```
raggio = float(input('Inserisci il raggio del cerchio')).
```

Per verificare il mio codice con Dcoder è perfettamente inutile il messaggio di richiesta dell'input, in quanto non sarebbe gestito, e basta scrivere

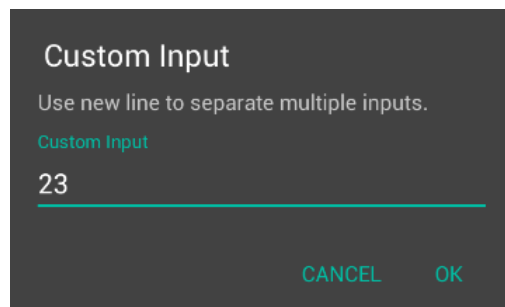
```
raggio = float(input()).
```

Occorre però attivare l'input e inserire il dato attraverso la relativa finestra prima della compilazione.

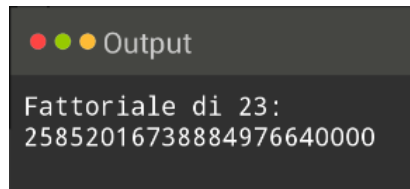
Fedeli al linguaggio Python vediamo questo codice che calcola il fattoriale di un numero con un magnifico algoritmo ricorsivo

```
Python 3 Input
1 def fatt(n):
2     if(n == 0) or (n == 1):
3         return 1
4     else:
5         return n*fatt(n-1)
6 n = int(input())
7 f = fatt(n)
8 print('Fattoriale di ' + str(n) + ":")
9 print(f)
```

Vediamo l'input attivato (con il pallocco azzurro spostato a destra) e in conseguenza di ciò inseriamo il numero di cui vogliamo calcolare il fattoriale in questa finestra



Dato l'OK avviamo la compilazione e troveremo questo output



I dati da inserire possono essere più di uno e richiamati in momenti diversi.

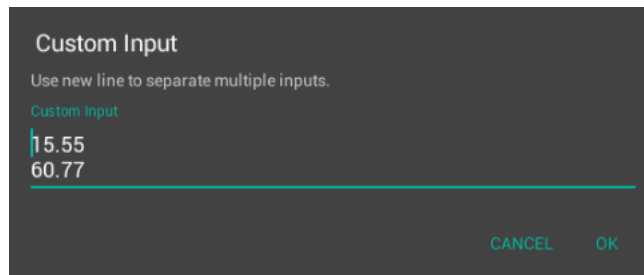
In questo caso essi si inseriscono nella finestra di input, nell'ordine in cui vengono chiamati nel codice, su righe diverse.

Abbiamo qui, per esempio, un codice scritto nel linguaggio C++ che calcola la somma di due numeri.

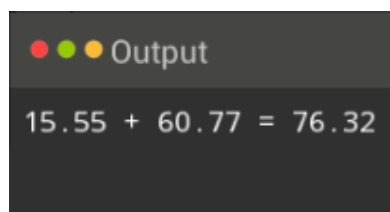
```
C++
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     double a, b, c;
6     cin >> a;
7     cin >> b;
8     c = a + b;
9     cout << a << " + " << b << " = " << c;
10    return 0;
11 }
```

Vediamo due inserimenti, anche qui non supportati da dialogo, per le variabili a e b, che sono il primo e il secondo addendo.

Per soddisfare queste due richieste, una dopo l'altra, compileremo così la finestra dell'input



e otterremo questo risultato



Può darsi, infine, che i dati da inserire debbano andare a formare un array.

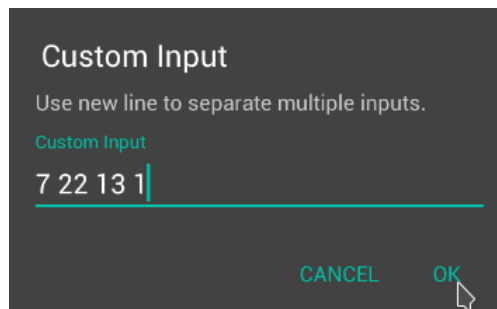
In questo caso abbiamo due possibilità: se l'istruzione di lettura per l'input si riferisce ad una riga, inseriamo i dati sulla stessa riga, separati da uno spazio; se l'istruzione di lettura per l'input cambia riga per ogni dato, inseriamo i dati uno per riga.

Questo codice, scritto, tanto per cambiare, in linguaggio Pascal, individua il valore massimo tra alcuni valori (4 nell'esempio) inseriti dall'utente.

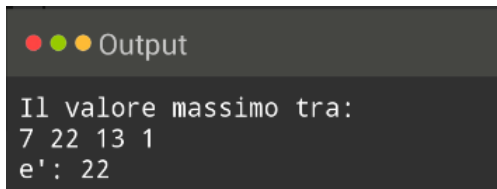
```
Pascal Input
1 program massimo;
2 var
3     i, max: integer;
4     a: array[1..4] of integer;
5 begin
6     for i := 1 to 4 do begin
7         read(a[i]);
8     end;
9     max := a[1];
10    for i := 2 to 4 do
11        if a[i] > max then
12            max := a[i];
13    writeln('Il valore massimo tra:');
14    for i := 1 to 4 do begin
15        write(a[i], ' ');
16    end;
17    writeln();
18    writeln('e': ', max)
19 end.
```

L'istruzione deputata a introdurre i dati, vediamo, è read, che, nel ciclo for, legge i dati senza cambiare riga.

Allora l'input sarà scritto così



E, con la compilazione, otterremo questo risultato



Se l'istruzione deputata a introdurre i dati fosse stata `readln`, che, nel ciclo `for`, leggerebbe i dati cambiando riga, avremmo dovuto scrivere i dati di input non sulla stessa riga ma ciascuno su una riga diversa.