



# **Programmare è FACILE con BAS.I.LI.CO.**

Di Massimo Messina

<https://corsofacile.blogspot.com>

Massimo Messina  
[smassimomessina@gmail.com](mailto:smassimomessina@gmail.com)  
Pisa, Italia

Liberamente si può condividere - copiare, distribuire e trasmettere l'opera alle seguenti condizioni:

- va attribuita l'opera o qualsiasi frammento dell'opera all'autore (ma in nessun modo che suggerisca che egli approvi te o il tuo utilizzo dell'opera),
- non si può usare quest'opera per scopi commerciali,
- modificando, trasformando o sviluppando quest'opera, si può distribuire l'opera risultante solo con la stessa licenza o con una licenza simile ad essa.

## Sommario - Da inserire

Un grande ringraziamento va a Rob Galleon, che ha creato il QB64 negli anni 2000. Ringrazio calorosamente anche mia figlia, con la quale fin da piccola ho condiviso la mia passione per l'informatica, e Gabriele Gessati, che è stato, pur se per un breve periodo, mio studente appassionato di programmazione informatica (in particolare di BBC BASIC) nel corso F.A.C.I.L.E. tenuto da me e da mia figlia a Pisa, corso che continua adesso anche nel web nel seguente blog:

<http://corsofacile.blogspot.com>

Sono state usate clip art di pubblico dominio tratte dal seguente sito:

<http://www.openclipart.com>

# Prefazione

Questo libro intende essere un'introduzione alla programmazione per principianti di ogni età, a partire dai ragazzi delle scuole medie. Intende mostrare come si possano usare i computer come strumenti attraverso i quali esprimere creatività, che possono aiutare a sviluppare la capacità di risolvere problemi, che poi è l'essenza della programmazione.

Ricordo la prima volta che ebbi tra le mani un computer: ero un bambino all'inizio degli anni '80 e si trattava di un Commodore 64, che mio padre vendeva nell'ambito della sua attività di agente librario. Uno dei canali di diffusione degli home computer, infatti, fu quello di abbinarlo ad enciclopedie che permettevano agli utenti di studiare quell'aggeggio che avevano comprato.

Non tutti negli anni '80 comprarono il computer insieme ad un'enciclopedia, ma il computer era comunque corredato da un manuale. Il manuale del Commodore 64 spiegava bene i rudimenti della programmazione con il linguaggio di programmazione BASIC (nella scarna, essenziale versione del linguaggio per Commodore 64) e vi era l'elenco completo dei comandi con relativa spiegazione della loro sintassi ed utilizzo.

L'home computer si presentava come una tastiera che si collegava al televisore attraverso il cavo dell'antenna. Ciò che veniva scritto sulla tastiera compariva sullo schermo.

Cosa iniziavano a fare, quindi, i bambini con il computer di casa? Quando iniziavamo a capire che non si trattava solamente di un dispositivo per far apparire sullo schermo della TV ciò che digitavamo sulla tastiera si cercava il manuale e lo si leggeva per provare man mano le righe di codice che venivano proposte. Da lì a modificarle per vedere che succedeva il passo era breve e così la creatività portava a realizzare i primi listati che facevano fare al computer ciò che volevamo noi.

Non tutti usarono così gli home computer. Moltissimi li usarono esclusivamente come console di videogiochi. Tutti però avevamo ben chiaro che un computer non faceva nulla senza il software, senza i programmi, senza le cassette (e poi i floppy disk).

Oggi la situazione è profondamente mutata. Già da bambini moltissimi hanno accesso a computer, smartphone o tablet, oltre che a console di videogiochi. Imparano presto a capire che tutti questi dispositivi hanno bisogno di software, ma ora il software è (a pagamento o meno) disponibile in rete e quindi è molto più semplice averlo. Immaginare un computer come qualcosa che non fa nulla senza essere programmato è molto più difficile oggi, per l'utente medio. Si finisce per vedere un computer come strumento ludico-multimediale e di comunicazione. Il rischio è di farsi coinvolgere da mondi virtuali nei quali, senza le dovute

precauzioni, i minori vengono in qualche modo programmati piuttosto che programmare i computer.

L'unico linguaggio di programmazione per chi aveva appena comprato il computer negli anni '80 era quello preinstallato in ROM, immediatamente disponibile già all'accensione (istantanea) del computer: il BASIC. L'acronimo sta per "Beginner's All-purpose Symbolic Instruction Code" (in italiano "codice di istruzione simbolica di uso generale per principianti"), ma la parola «basic» in sé in inglese significa «di base», «fondamentale», «essenziale» e quel nome è quindi tutto un «programma».

Non passò troppo tempo che altri linguaggi furono disponibili. Uno di questi fu il Logo, linguaggio di programmazione fortemente orientato alla grafica, alla geometria di base ed alla didattica, adatto all'insegnamento della programmazione a partire dai bambini della scuola primaria. Una sua implementazione open source per i computer attuali è FMSLogo, tradotto in 12 lingue, tra le quali l'italiano.

Il Logo nacque negli anni '60, sviluppato da Seymour Papert del Massachusetts Institute of Technology. Papert aveva lavorato con Jean Piaget all'Università di Ginevra dal 1958 al 1963 ed usò, nello sviluppo del Logo, i risultati degli studi di Piaget, notissimo psicologo, biologo, pedagogista e filosofo svizzero che fondò l'epistemologia genetica e si occupò di psicologia dello sviluppo.

La caratteristica peculiare del Logo è la geometria della tartaruga. In origine il Logo fu usato per far muovere un semplice robot, che aveva una corazza la cui forma faceva pensare ad una tartaruga. Con lo sviluppo dei monitor il Logo poté essere usato anche in assenza del robot ed entrò così nei laboratori informatici delle scuole, particolarmente per lo studio della geometria.

La tartaruga, non essendo più un robot, è uno sprite che può avere la forma di un triangolo o un disegno di una tartaruga stilizzata (un triangolo). Chi usa il Logo può impartire comandi di movimento alla tartaruga, che muovendosi realizzerà dei disegni sullo schermo.

La grafica della tartaruga si contraddistingue per il suo modo di impartire i comandi di disegno al computer descrivendoli "dall'interno", cioè immedesimandosi nella tartaruga. Dal punto di vista pedagogico questo modo di disegnare attraverso comandi dati al computer è strettamente legato all'esperienza diretta di movimento dell'utente ed alla capacità di comunicare agli altri come muoversi nello spazio.

Il Logo è stato usato nelle scuole elementari e medie inferiori, dato che, attraverso i suoi comandi, anche un principiante può fin dall'inizio ottenere risultati visibili. Vi sono versioni di Logo con comandi in varie lingue, tra le quali alcune versioni italiane, fin dagli anni '80. Il Logo ha anche il vantaggio, rispetto al BASIC, di imporre un metodo di programmazione ordinato, strutturato, usando le procedure ed essendo così estensibile.

Una delle implementazioni per i computer di oggi del linguaggio Logo è FMSLogo. Il sito di FMSLogo è il seguente:

<https://fmslogo.sourceforge.io>

Per usare FMSLogo in italiano si può partire dalle seguenti pagine web:

<https://entuland.com/it/libri/fmslogo>

<https://www.tiziana1.it/logo.htm>

In questo libro, invece, si parlerà del più antico linguaggio di programmazione per principianti: il BASIC, in una versione recente: il BAS.I.LI.CO. (BASato sull'Italiano Linguaggio COmpatibile). Ho realizzato BAS.I.LI.CO. derivandolo dal QB64. BAS.I.LI.CO. è un'implementazione del BASIC con istruzioni basate sulla lingua italiana, è multiplatforma, completamente gratuito ed a codice aperto. Si può usare su sistemi Windows, MacOS e Linux. Lo ritengo adatto alla didattica informatica, come lo era il BASIC negli anni '80, per i ragazzi delle scuole medie e superiori, ma anche per i bambini della scuola primaria e anche per i principianti assoluti di ogni età.

BAS.I.LI.CO. si scarica da qui:

<https://github.com/smaxmex/basilico>

Manca ancora documentazione in italiano e cerco di colmare tale lacuna anche con questo libro, che è un percorso di lezioni, corredate di esercizi, per insegnare la programmazione a chiunque voglia cimentarsi in questa meravigliosa avventura. Le lezioni di questo libro si basano sul libro intitolato "So You Want to Learn to Program?" ("Così vuoi imparare a programmare?") di James M. Reneau, libro in varie lingue, ma non in italiano, che insegna a programmare in BASIC.

Le informazioni relative al libro di Reneau sono al seguente sito in inglese:

<http://www.basicbook.org>

BASIC-256 si può scaricare dalla seguente pagina:

<https://downloads.sourceforge.net/project/kidbasic>

<https://github.com/smaxmex/basilico>

# Capitolo 1: Ecco BAS.I.LI.CO. - Ciao, mondo!

Questo capitolo introdurrà l'Ambiente di Sviluppo di BAS.I.LI.CO. usando l'istruzione **Stampa**. Si vedrà la differenza tra i comandi inviati al computer, le stringhe di testo e i numeri che saranno usati dal programma. Esploreremo anche la matematica semplice per mostrare quanto sia talentuoso il tuo computer. Infine, si imparerà cos'è un errore di sintassi e come correggerlo.

## La finestra di BAS.I.LI.CO.

La finestra di BAS.I.LI.CO., chiamata anche Ambiente di Sviluppo, è divisa in tre sezioni: barra dei menu, area del programma e area dello stato (qui di seguito nella Figura 1 vediamo l'Ambiente di Sviluppo di BAS.I.LI.CO.).

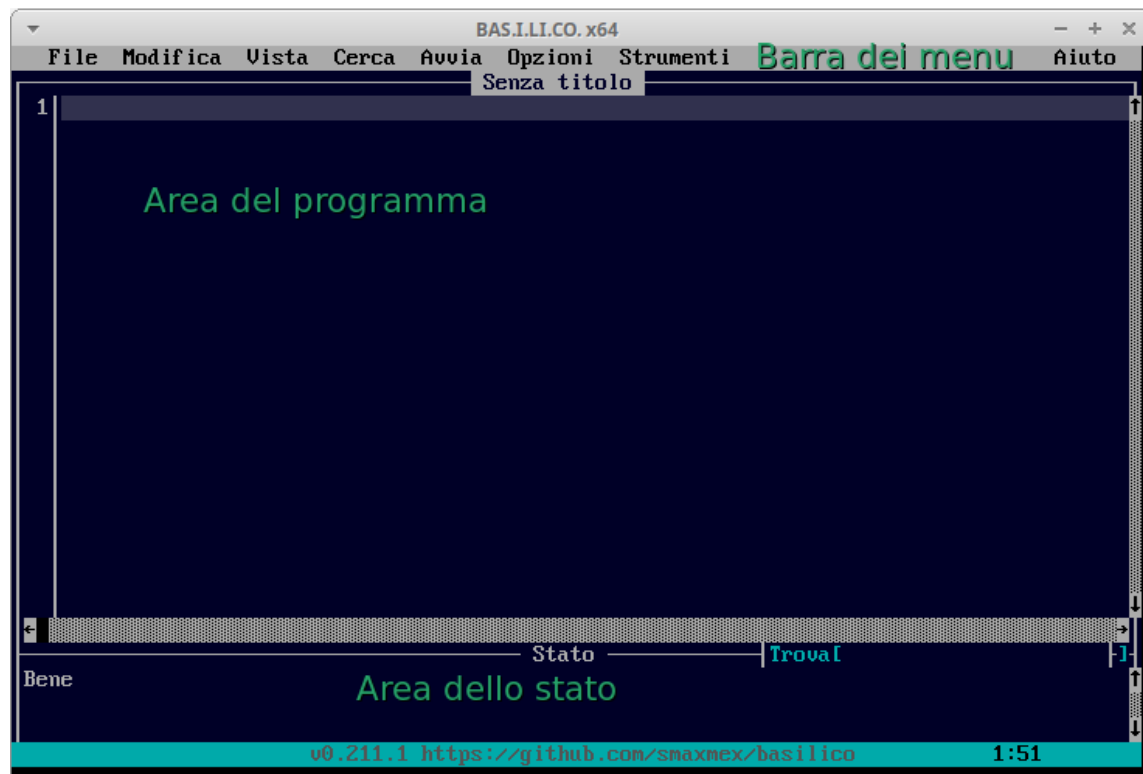


Figura 1: Ambiente di Sviluppo di BAS.I.LI.CO.

### Barra dei menu

La barra dei menu contiene diversi menu a discesa. Questi menu includono: "File", "Modifica", "Vista", "Cerca", "Avvia", "Opzioni", "Strumenti" e "Aiuto". Il menu "File" consente di iniziare un nuovo programma, caricare programmi salvati, salvare il programma attuale e uscire dall'Ambiente di Sviluppo. Il menu "Modifica"



consente di annullare l'ultima modifica al programma, ripetere l'ultima modifica appena annullata, tagliare, copiare, incollare, cancellare e selezionare testo nell'area del programma, commutare in commento una linea del programma, aggiungere o rimuovere all'inizio di una linea del programma l'apostrofo rendendola commento, creare nuova SUB o nuova FUNZIONE (vedremo più avanti cosa sono i commenti, le SUB e le FUNZIONI). Il menu "Vista" consente di mostrare l'elenco di SUB e FUNZIONI, mostrare o nascondere i numeri delle linee del programma, mettere un colore di sfondo a tali numeri, mostrare o meno un separatore tra i numeri di linea e il programma, mostrare gli avvisi del compilatore. Il menu "Cerca" consente di cercare stringhe ed eventualmente sostituirle nel programma, di cancellare la cronologia delle ricerche, di attivare e disattivare la navigazione rapida nell'area del programma, di aggiungere/rimuovere segnalibri nella stessa, di andare avanti e indietro nei segnalibri e di andare ad un specifico numero di linea. Il menu "Avvia" permette di compilare e di eseguire il programma. I menu "Opzioni" e "Strumenti" contengono rispettivamente opzioni relative all'ambiente di sviluppo e strumenti utili riguardanti i codici dei caratteri, le espressioni matematiche e i colori. Il menu "Aiuto" visualizza attualmente informazioni sulle istruzioni del QB64, dal quale deriva BAS.I.LI.CO., e sulla versione di BAS.I.LI.CO. che si sta usando. Le istruzioni di BAS.I.LI.CO. derivano da quelle del QB64. Chi non ha problema con l'inglese può così conoscere la sintassi di ogni istruzione di BAS.I.LI.CO. vedendo nel file "Tabella di derivazione dal QB" (il cui contenuto riporto qui di seguito) a quale istruzione in inglese corrisponde ogni istruzione in italiano e poi cercare spiegazione e sintassi per ogni istruzione in inglese, usando il menu "Aiuto" o le pagine di guida ai comandi dei siti sul QB64. La seguente pagina mi sembra la migliore tra esse:

[https://qb64phoenix.com/qb64wiki/index.php/Keyword\\_Reference\\_-\\_Alphabetical](https://qb64phoenix.com/qb64wiki/index.php/Keyword_Reference_-_Alphabetical)

### **Tabella di derivazione dal QB**

• Print	Stampa
• GoTo	Vai
• Cls	Psc
• Rem	Nota
• Not	Non
• And	E
• Or	O
• Xor	Ox
• Let	Sia
• Input	Lettura
• For	Per
• To	A
• Step	Passo
• Next	Seguente
• If	Se
• Then	Allora

<https://github.com/smaxmex/basilico>

• Else	Altr
• ElseIf	AltrSe
• End	Fine
• Stop	Stop
• Color	Colore
• Run	Avvia
• Sleep	Dormi
• Right\$	Destra\$
• Left\$	Sinistra\$
• GoSub	VaiSub
• Return	Torna
• Function	Funzione
• Call	Chiama
• Dim	Dim
• Command\$	Comando\$
• Read	Leggi
• Data	Dati
• Screen	Schermo
• Pset	Pixel
• Restore	Rileggi
• Line	Linea
• Circle	Cerchio
• Paint	Tingi
• Beep	Bip
• Sound	Suono
• As	Come
• Locate	Cursore
• Abs	Ass
• Rnd	Cas
• Int	Int
• Point	Punto
• On...	Al...
• Stop	Stop
• Tab	Tab
• Wait	Aspetta
• Poke	Poni
• Peek	Vedi
• Clear	Pulisci
• System	Sistema
• InKey\$	Tasto\$
• Spc	Spz
• Sgn	Sgn
• Pos	Pos
• Sqr	Rqu
• Log	Log
• Exp	Esp
• Sin	Sen
• Cos	Cos
• Tan	Tan

• Atn	Atn
• Len	Lun
• Str\$	Str\$
• Val	Val
• Asc	Asc
• Chr\$	Car\$
• Mid\$	Mez\$
• Open	Apri
• Close	Chiudi
• Input\$	Lettura\$
• Output	Scrittura
• Write	Scrivi
• Times	Orario\$
• Date\$	Data\$
• Get	Prendi
• Put	Metti
• Randomize	Casualizza
• Timer	Tempo
• On	Acceso
• Off	Spento
• Const	Cost
• Error	Errore
• Resume	Riprendi
• Using	Usando
• EOF	FDF
• LOC	LOC
• Mod	Mod
• Do	Fai
• Loop	Ciclo
• Until	Fino
• While	Mentre
• Wend	Mfine
• Exit	Esci
• Erase	Cancella
• Integer	Intero
• Long	Lungo
• Single	Singola
• Double	Doppia
• String	Stringa
• Unsigned	Positivo
• String\$	Stringa\$
• Eqv	Eqv
• Imp	Imp
• DEFINT	DEFINT
• DEFLNG	DEFLNG
• DEFSNG	DEFSNG
• DEFDBL	DEFDPP
• DEFSTR	DEFSTR
• Swap	Scambia

• Lprint	LStampa
• Space\$	Spazi\$
• Declare	Dichiara
• Library	Libreria
• Dynamic	Dinamica
• CustomType	Personalizzata
• Static	Statica
• ByVal	PerVal
• Type	Tipo
• Alias	Alias
• Random	Casuale
• Binary	Binario
• Append	Aggiunta
• Shared	Condiviso
• Access	Accesso
• Lock	Blocca
• Unlock	Sblocca
• LOF	LDF
• Seek	Cerca
• Com	Com
• Option	Opzione
• Base	Base
• Ubound	LimiteS
• Lbound	LimiteI
• Redim	Ridim
• Shell	Esegui
• Csrln	Csrln
• Inp	Let
• Out	Scr
• Key	Tasti
• Lpos	Lpos
• View	Vista
• Width	Misure
• Palette	Tavolozza
• PCopy	CopiaP
• PMap	Piano
• Window	Finestra
• Preset	Cpixel
• ChDir	Posizione
• Kill	Elimina
• MkDir	Crea
• Name	Rinomina
• RmDir	Rimuovi
• FileAttr	Attributi
• Draw	Disegna
• Play	Suona
• FreeFile	Libero
• Hex\$	Esa\$
• Oct\$	Ott\$

• InStr	InStr
• Lcase\$	Minuscolo\$
• Ucase\$	Maiuscolo\$
• LSet	MettiS
• RSet	MettiD
• LTrim	TogliS
• RTrim	TogliD
• CDb1	DoPV
• CInt	InPV
• CLng	LuPV
• CSng	SiPV
• CVDMBF	CVDFBM
• MKDMBF	CrDFBM
• CVSMBF	CVSFBM
• MKSMBF	CrSMBF
• ErDev	ErDis
• ErDev\$	ErDis\$
• ErL	ErL
• Pen	Penna
• STrig	Pulsanti
• Stick	Manopola
• Absolute	Assoluto
• Any	Ogni
• Bload	Carica
• Bsave	Salva
• VarPtr	VarPtr
• VarSeg	VarSeg
• Chain	Passa
• CVD	CVD
• CVI	CVI
• CVL	CVL
• CVS	CVS
• DEF	DEF
• SEG	SEG
• Environ	Ambiente
• Environ\$	Ambiente\$
• Is	Risulta
• Field	Campo
• Files	Lista
• Fix	Mozza
• MKD\$	CrD\$
• MKI\$	CrI\$
• MKL\$	CrL\$
• MKS\$	CrS\$
• Common	Comune
• Interrupt	Interruzione
• InterruptX	InterruzioneX
• SAdd	SAdd

## Area del programma

I programmi sono composti da istruzioni che dicono al computer esattamente cosa fare e come farlo. Digiterai i tuoi programmi, modificherai e correggerai il tuo codice e caricherai i programmi salvati in quest'area dello schermo.

## Area dello stato

In quest'area compaiono gli avvisi e i messaggi di errore, non ancora tradotti in italiano dal QB64.

## Il tuo primo programma - L'istruzione Stampa

Scriviamo effettivamente un programma per computer. Vediamo se BAS.I.LI.CO. ci saluterà. Nell'area del programma digita il seguente programma di una linea (vedrai il numero di linea in BAS.I.LI.Co. senza digitarlo tu):



```
1 | Stampa "Ciao, mondo!"
```

Programma 1: Saluta

Di solito trovate nei manuali di programmazione il virgolettato sopra nella sua versione inglese: "Hello, world!". Dopo aver digitato il programma, cliccate su "Avvia" e poi su "Inizia". BAS.I.LI.CO. compilerà ed eseguirà il programma.

 <b>Nuovo Concetto</b>	<b>Stampa espressione</b> L'istruzione <b>Stampa</b> viene usata per far sì che BAS.I.LI.CO. stampi un'espressione nella finestra di esecuzione del programma.
--	---

BAS.I.LI.CO. tratta lettere, numeri e punteggiatura che si trovano all'interno delle virgolette come un blocco. Questo blocco è chiamato stringa.

 <p><b>Nuovo concetto</b></p>	<p>"lettere, numeri 9988 e simboli &amp;%"</p> <p>Una stringa inizia con le virgolette doppie (") e terminare allo stesso modo.</p>
 <p><b>Nuovo concetto</b></p>	<p>Cliccare "Avvia" e poi "Inizia" e BAS.I.LI.CO. compilerà ed eseguirà il programma.</p> <p>Devi dire a BAS.I.LI.CO. quando vuoi che inizi a eseguire un programma. Non sa automaticamente quando si è finito di digitare il codice di programmazione. Si può fare cliccando "Avvia" e poi "Inizia". Al posto di questa sequenza si può premere il tasto F5.</p>

Per cancellare il programma su cui stai lavorando e avviare completamente un nuovo programma, si clicca sul menu "File" per poi cliccare su "Nuovo". visualizzerà la seguente finestra di dialogo:

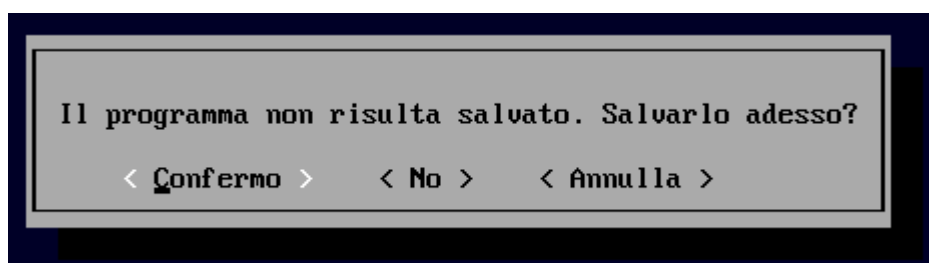



Figura 2 : BAS.I.LI.CO. - Finestra di dialogo per nuovo programma

Se sei d'accordo con la cancellazione del programma dallo schermo, clicca "Confermo". Se accidentalmente hai cliccato su "Nuovo" e non vuoi iniziare un nuovo programma, clicca su "Annulla".

 <p><b>Nuovo concetto</b></p>	<p>"File" e poi "Nuovo" nel menu</p> <p>Il comando "Nuovo" dice a BAS.I.LI.CO. che vuoi cancellare le istruzioni correnti dall'area del programma e avviare un programma completamente nuovo. Se non hai salvato il tuo programma sul computer (Capitolo 2), perderai tutte le modifiche che hai apportato al programma.</p>
--	--


## Secondo programma - Stampare qualcos'altro

Puoi anche far sì che l'istruzione **Stampa** esprima numeri. Prova il seguente programma:

```
1 | Stampa 123456789
```

Programma 2: Stampa un numero

Dopo aver digitato il programma, premi F5 per compilarlo ed eseguirlo. BAS.I.LI.CO. ha scritto ciò che ti aspettavi?

	<p><b>Numeri</b></p> <p>BAS.I.LI.CO. consente di immettere numeri in formato decimale. Non utilizzare il punto quando si immettono numeri grandi. Il punto serve per i numeri decimali. Per i numeri inferiori a zero, basta mettere il segno meno prima del numero. Per numeri con cifre decimali dopo la virgola inferiori a 1, lo zero si può omettere ed in ogni caso non verrà stampato nella finestra di esecuzione del programma.</p> <p>Esempi: 1.56, 23456, -6.45 e .5</p>
---	---

## BAS.I.LI.CO. è davvero bravo con i numeri - Aritmetica semplice

Il cervello del computer (chiamato unità di elaborazione centrale, colloquialmente nota semplicemente come processore o con l'acronimo inglese CPU, che sta per Central Processing Unit) lavora esclusivamente con numeri. Tutto ciò che fa, dalla grafica al suono e tutto il resto, viene fatto manipolando i numeri. Le quattro operazioni di base di addizione, sottrazione, moltiplicazione e divisione vengono eseguite utilizzando gli operatori indicati nella Tabella 1.

Operatore	Operazione	Esempio
+	Somma	espressione1 + espressione2
-	Sottrazione	espressione1 - espressione2
*	Moltiplicazione	espressione1 * espressione2
/	Divisione	espressione1 / espressione2

Tabella 1: Operatori matematici di base



Prova il seguente programma.

```
1 | Stampa 12 * (2 + 10)
```


Programma 3: Stampa il risultato

Il computer dovrebbe dare come risultato "144".

```
1 | Stampa 5 / 2
```

Programma 4: Stampa altro risultato

Il computer ha dato come risultato "2.5"?

	<p><b>Nuovo concetto</b></p> <p><b>+</b> <b>-</b> <b>*</b> <b>/</b> <b>()</b></p> <p>Le quattro operazioni matematiche di base: addizione (+), sottrazione (-), divisione (/) e moltiplicazione(*) funzionano con i numeri per eseguire calcoli. Un valore numerico è richiesto su entrambi i lati di questi operatori. Puoi anche usare le parentesi per raggruppare le operazioni.</p> <p>Esempi: <math>1 + 1</math>, <math>5 * 7</math>, <math>3.14 * 6 + 2</math>, <math>(1 + 2) * 3</math> e <math>5 - 5</math></p>
--	--

## Concatenazione

La concatenazione è l'operazione che unisce due stringhe insieme per creare una stringa più lunga. Se le stringhe "abcd" e "xyz" fossero concatenate insieme, il risultato sarebbe la stringa "abcdxyz". Questa operazione è chiamata concatenazione. L'operatore **;** concatena numeri e stringhe nel comando **Stampa**, l'operatore **+** sommerà numericamente due numeri, ma li concatenerà se sono stringhe (vedremo in seguito).

Proviamo:

```
1 | Stampa "Ciao, "; "Maria."
```

## Programma 5: Saluta Maria

Il computer dovrebbe salutare Maria.

Provane un altro.


```
1 | Stampa 2; "altre volte"
```


## Programma 6: Stampa altro

Nell'ultimo esempio la concatenazione è stata eseguita con un numero e una stringa.

```
1 | Stampa 1 + 2
2 | Stampa 1 ; 2
```

Il computer dovrebbe dare come risultato 3 e 1 2. Nella prima riga, l'operatore più somma i numeri 1 e 2. Nella riga 2 l'operatore punto e virgola concatena le cifre, ma lasciando uno spazio tra esse.

 <b>Nuovo concetto</b>	<b>; (concatenare)</b>
	<b>+ (concatenare)</b>  Il punto e virgola (;) viene utilizzato per dire al computer di concatenare (unire) le stringhe.  L'operatore + esegue la somma di numeri e la concatenazione di stringhe.

 <b>Nuovo concetto</b>	<b>Stampa espressione</b>
	<b>Stampa espressione;</b>  L'istruzione <b>Stampa</b> viene utilizzata per visualizzare testo e numeri nella finestra di esecuzione del programma. L'istruzione Stampa fa sì che la stampa successiva andrà sulla riga successiva. Se si inserisce un ; (punto e virgola) alla fine dell'espressione, verrà soppresso l'avanzamento di riga così che la stampa successiva sarà sulla stessa riga.

```

1 | Psc
2 | Stampa "Ciao, ";
3 | Stampa "amico ";
4 | Stampa "mio."

```

Programma 8: Molte stampe sulla stessa riga



**Nuovo  
concetto**

### **Psc**

L'istruzione **Psc** cancella l'intera area della finestra di esecuzione.

## **Che cosa è un "Errore di sintassi"**

I programmatori sono umani, commettono errori. Gli "errori di sintassi" sono uno dei tipi di errori che potremmo incontrare. Un "errore di sintassi" viene generato da BAS.I.LI.CO. quando non capisce il programma che hai digitato. Di solito gli errori di sintassi sono causati da errori di ortografia, virgole mancanti, spazi non corretti, virgolette o parentesi non chiuse. BAS.I.LI.CO. ti dirà su quale riga si trova l'errore.

## Esercizi



**Cerca  
le parole**

p s c d s t a m p a e m c  
j g j r o i q l o n q o x  
a v v i a u n i o c n s z  
v w s y o b s i k c i l l  
e n a t i s s p a n p a x  
r s e p e s r t t f r p t  
r b k r e o e a r m m r a  
o r p r g n s d o i f n i  
r x p r a s t i h l n a f  
e s a r i d w n v e d g i  
e m e a c v c e i j f d a  
v i r g o l e t t e c i s  
n a m z p r o g r a m m a

Psc, concatenare, errore, espressione, stampa, programma, virgolette, avvia, stringa, sintassi



**Problemi**

1. Scrivi un programma di una riga che stampi lo scioglilingua "Sopra la panca la capra campa, sotto la panca la capra crepa".
2. Usa il computer come calcolatrice per risolvere il seguente problema e dire la risposta: Roberto ha 5 caramelle e Giacomo ne ha 9. Se dovessero dividere le caramelle equamente tra loro, quante ne avrebbe ciascuno (in media)?
3. Usa il computer come calcolatrice per risolvere il seguente problema e dire la risposta: vuoi 5 modellini di auto che costano ciascuno 1,25€ e un modellino di barca che costa 3,50€. Di quanti soldi hai bisogno per comprarli?
4. Scrivi un programma di una riga che scriva "uno più due uguale tre" senza usare la parola tre o il numero 3.

## Capitolo 2: Disegnare forme elementari

In questo capitolo ci occuperemo di grafica. Si potrà imparare a disegnare rettangoli, cerchi, linee e punti di vari colori. I programmi diventeranno sempre più complessi, quindi si imparerà anche come salvare i propri programmi in un archivio a lungo termine e come ricaricarli in modo da poterli eseguire di nuovo o modificarli.

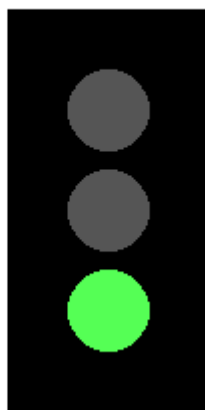
### Disegnare rettangoli e cerchi

Cominciamo la parte grafica scrivendo un programma che disegnerà un semaforo, in particolare un semaforo verde.

```
1 ' semaforo.bas
2 ' Mostra un semaforo e scrive un messaggio
3 Schermo 12
4 Colore 0, 15
5 Psc
6 Linea (100, 50)-(200, 250), , RP
7 Colore 8
8 Cerchio (150, 100), 20
9 Tingi (159, 100)
10 Colore 8
11 Cerchio (150, 150), 20
12 Tingi (150, 150)
13 Colore 10
14 Cerchio (150, 200), 20
15 Tingi (150, 200)
16 Colore 0
17 Stampa "Semaforo verde. Puoi andare."
```


Programma 9: Semaforo

Semaforo verde. Puoi andare.



Programma 9 in esecuzione: semaforo

Esaminiamo riga per riga il programma. La prima e la seconda riga sono chiamate istruzioni di note o commenti. Una nota è un'istruzione attraverso cui il programmatore può inserire commenti nel suo codice che sono ignorati dal linguaggio di programmazione. Servono per descrivere cosa stanno facendo blocchi di codice o per inserire il nome del programma, per scrivere perché abbiamo scritto un programma o chi è il programmatore.


 <b>Nuovo concetto</b>	<p><b>Nota</b></p> <p>Le istruzioni <b>'</b> e <b>Nota</b> sono chiamate note. Un'istruzione di nota consente al programmatore di inserire commenti nel codice del programma su cui sta lavorando. Il computer vede l'istruzione <b>'</b> o <b>Nota</b> e ignorerà tutto il resto del testo sulla riga.</p>
--	---

Alla riga tre vediamo l'istruzione **Schermo** con il parametro 12, per impostare una modalità grafica dello schermo che ci permette di usare i comandi grafici delle righe seguenti **Linea**, **Cerchio** e **Tingi**. Alla riga quattro, con il comando **Colore**, viene impostato il colore di primo piano e il colore di sfondo, rispettivamente il primo e il secondo numero, separati da una virgola, del comando **Colore**. I numeri dei colori sono quelli della seguente tabella. Tra parentesi per ogni colore ci sono i corrispondenti valori RGB, che sono spiegati qui di seguito. È stato inserito nella tabella anche il colore arancione, che non è tra i 16 colori base di BAS.I.LI.CO, come esempio di colore ricavabile tramite la tripletta RGB.

0	Nero (0,0,0)	8	Grigio (128,128,128)
1	Blu scuro (0,0,128)	9	Blu (0,0,255)
2	Verde scuro (0,128,0)	10	Verde (0,255,0)
3	Ciano (0,128,128)	11	Acqua (0,255,255)
4	Rosso scuro (128,0,0)	12	Rosso (255,0,0)
5	Magenta scuro (128,0,128)	13	Magenta (255,0,255)
6	Giallo scuro (128,128,0)	14	Giallo (255,255,0)
7	Argento (164,164,164)	15	Bianco (255,255,255)
	Arancione scuro (170,51,0)		Arancione (255,102,0)


Tabella 2: Tabella dei 16 colori base + due tonalità di arancione

Alla riga cinque c'è l'istruzione **Psc**, che conosciamo già dal Capitolo 1.

 <b>Nuovo concetto</b>	<b>Colore</b> <i>,colore_di_sfondo%</i> <b>Psc</b>  L'istruzione <b>Psc</b> cancella la finestra di esecuzione del programma, lasciando uno spazio libero in cui poter disegnare.  Se prima del comando <b>Psc</b> viene impostato un colore di sfondo con il comando <b>Colore</b> , l'intera finestra di esecuzione verrà impostata su quel colore.
--	--

Le righe quattro, otto, dieci, tredici e sedici contengono l'istruzione **Colore**, per indicare a BAS.I.LI.CO. quale colore usare per la successiva azione di scrittura o disegno. Puoi scegliere i colori usando uno dei 16 numeri di colori standard oppure puoi crearne uno tra oltre 16 milioni di colori diversi mescolando insieme i colori primari (rosso, verde e blu), usando la funzione **\_RGB(rosso%,verde%,blu%)** al posto del valore numerico da 0 a 15 nel comando **Colore**.

Usando **\_RGB(rosso%,verde%,blu%)** per scegliere un colore, assicurarsi di usare valori tra 0 e 255 per ognuno dei tre colori primari. Zero (0) rappresenta nessuna luce di quel colore e 255 significa la massima luce per quel colore. Il bianco è fatto così da 255, 255, 255 (tutti i colori) mentre il nero è rappresentato da 0, 0, 0 (nessun colore). Questa rappresentazione numerica è nota come tripletta RGB.

 <b>Nuovo concetto</b>	<b>Colore</b> <i>colore_per_disegnare%[,colore_di_sfondo]</i> <b>Colore</b> <b>_RGB(rosso%, verde%, blu%)</b>  L'istruzione <b>Colore</b> consente di impostare il colore che verrà usato per disegnare dai successivi comandi di disegno. È possibile far seguire all'istruzione <b>Colore</b> un numero da 0 a 16, corrispondendo ad ogni numero un colore (vedere tabella dei colori della pagina precedente). È inoltre possibile specificare oltre 16 milioni di colori diversi utilizzando la funzione <b>_RGB()</b> specificando quanto rosso, verde e blu devono essere utilizzati, usando per ognuno dei tre colori base dei numeri interi da 0 a 255. Preimpostato come colore per disegnare è il colore bianco, essendo preimpostato il nero come colore di sfondo.
--	---

Il parametro 12 nel comando **Schermo** imposta una finestra di esecuzione del programma larga 640 pixel (x) e alta 380 pixel (y). Un pixel è il punto più piccolo che può essere visualizzato sul monitor del computer. L'angolo in alto a sinistra è l'origine (coordinate 0,0) e quello in basso a destra è avrà coordinate

639,479. Ogni pixel può essere rappresentato da due numeri, il primo (x) è la distanza dal lato sinistro della finestra e il secondo (y) rappresenta la distanza dalla parte alta della finestra. Questo modo di contrassegnare i punti è una variante di ciò che è noto ai matematici come Sistema di Coordinate Cartesiane.

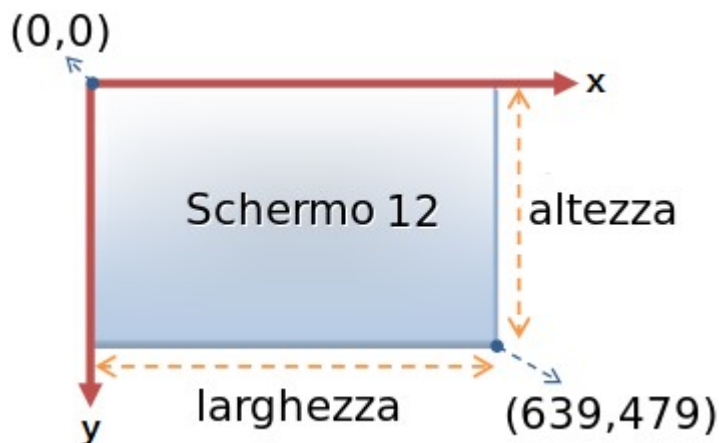


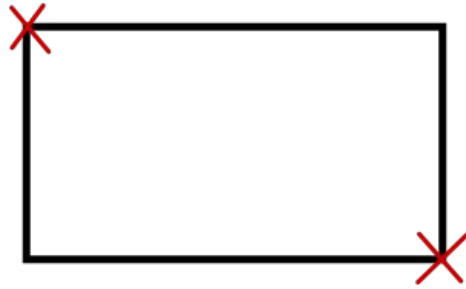
Illustrazione 4: Il sistema di coordinate cartesiane della finestra di esecuzione con il valore 12 del comando **Schermo**

Esaminiamo adesso l'istruzione **Linea**. Si può usare per disegnare segmenti o rettangoli sullo schermo. Nel programma che stiamo esaminando è usata per disegnare un rettangolo.

- Se vogliamo realizzare un segmento bisogna digitare dopo il comando **Linea** le coordinate dei punti estremi del segmento (essendo tali coordinate, come abbiamo visto sopra, una coppia di numeri tra parentesi), separando le coordinate dei due punti con un trattino (uguale al segno meno: "-"). Se vogliamo un colore specifico per la linea che vogliamo disegnare, possiamo indicarlo inserendo il numero del colore corrispondente, in base alla tabella che abbiamo visto sopra, dopo le coordinate dei due punti, mettendo una virgola tra coordinate dei punti e numero del colore. Nel caso del programma del semaforo è assente il parametro relativo al colore, usando così il colore stabilito dal comando **Colore** nella riga 4.
- Se vogliamo che il comando **Linea** faccia un rettangolo piuttosto che una linea, va aggiunta, sempre dopo un'ulteriore virgola la lettera R e se vogliamo che il rettangolo sia pieno, le lettere saranno due: RP. Va sempre tenuto in considerazione che tutti e quattro i valori numerici del comando **Linea** sono espressi in pixel (la dimensione del punto più piccolo che può essere visualizzato).



(x,y) di un angolo



(x,y) dell'angolo opposto

Illustrazione 7: Rettangolo

Come si può vedere, il rettangolo nel programma inizia nel punto con coordinate 100,50 ed è largo 100 pixel e alto 200 pixel.



**Nuovo  
concetto**

**Linea** (x1, y1) - (x2, y2)[, colore%, RP]

L'istruzione **Linea** usa il colore di disegno corrente (a meno che non si specifichi altrimenti, inserendo un valore numerico corrispondente ad un colore tra le due virgole). Il comando Linea serve sia per disegnare segmenti che per disegnare rettangoli. Per creare un rettangolo va messa la lettera R, se vogliamo che sia un rettangolo pieno, va aggiunta la lettera P. **Linea** fa parte dei comandi grafici che funzionano con modalità del comando **Schermo** che permettano tali comandi grafici, come Schermo 12. Nel caso del rettangolo (R) le due coppie di coordinate sono quelle relative a due angoli opposti del rettangolo che vogliamo che il computer disegni.

Le righe 8, 11 e 14 del Programma 9 introducono l'istruzione **Cerchio** per disegnare cerchi. Richiede tre argomenti numerici, i primi due rappresentano le coordinate cartesiane del centro del cerchio e il terzo il raggio in pixel.

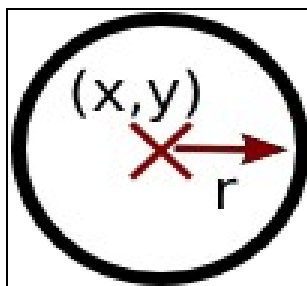


Illustrazione 8: Cerchio



**Nuovo  
concetto**

### **Cerchio ( $x$ , $y$ ), $r$**

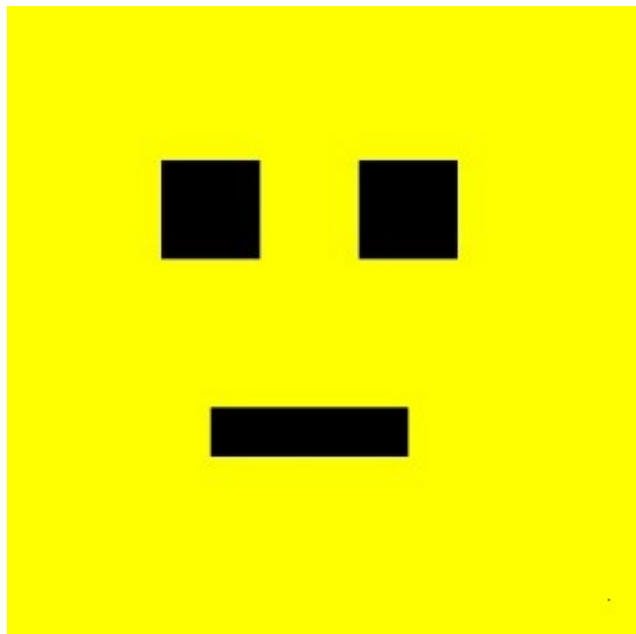
L'istruzione **Cerchio** utilizza il colore di disegno corrente e disegna un cerchio con centro in nel punto con coordinate  $x$  e  $y$  e con raggio  $r$ .

## Altri programmi che utilizzano cerchi e rettangoli

Ecco un paio di programmi di esempio che usano le istruzioni **Psc**, **Colore**, **Linea** e **Cerchio**. Digita i programmi e modificali. Trasformali in una faccia accigliata, in una faccia di un alieno o somigliante a qualcuno che conosci.

```
1 ' viso.bas
2
3 Schermo 12
4
5 ' rendi lo schermo giallo
6 Colore , 14
7 Psc
8
9 ' disegna la bocca
10 Colore 0
11 Linea (200, 200)-(300, 225), , RP
12
13 ' metti gli occhi
14 Linea (175, 75)-(225, 125), , RP
15 Linea (275, 75)-(325, 125), , RP
16
17 Stampa "Ciao"
```

Programma 10: Faccia con rettangoli



Esempio di esecuzione del programma 10: faccia con rettangoli

```

1 ' sorrisocerchio.bas
2
3 Schermo 12
4 ' pulisci lo schermo
5 Colore , 15
6 Psc
7 ' disegna il viso
8 Colore 14
9 Cerchio (350, 150), 150
10 Tingi (350, 150)
11 ' disegna la bocca
12 ' facendo un grande cerchio nero
13 ' e poi coprendo una parte
14 ' per formare un sorriso
15 Colore 0
16 Cerchio (350, 200), 70
17 Tingi (350, 200)
18 Colore 14
19 Cerchio (350, 150), 70
20 Tingi (350, 150)
21 ' disegna gli occhi
22 Colore 0
23 Cerchio (300, 100), 30
24 Tingi (300, 100)
25 Cerchio (400, 100), 30
26 Tingi (400, 100)

```

Programma 11: Volto sorridente con cerchi



Esempio di esecuzione del programma 11: volto sorridente con cerchi

## Salvataggio del programma e ricaricamento

Ora che i programmi stanno diventando più complessi, potresti volerli salvare in modo da poterli ricaricare in futuro. È possibile salvare un programma premendo contemporaneamente Ctrl e S o scegliendo l'opzione "Salva" nel menu File (usando il mouse o premendo prima Alt+F e poi Alt+S). Verrà visualizzata una finestra di dialogo che chiederà il nome che vogliamo dare al file, se si tratta di un nuovo programma, oppure salverà le modifiche apportate (sostituendo il vecchio file).

Se non si desidera sostituire la vecchia versione del programma e si desidera memorizzarla con un nuovo nome, è possibile utilizzare l'opzione "Salva come..." nel menu File per salvare una copia con un nome diverso. Per caricare un programma salvato in precedenza, premere contemporaneamente Ctrl e O o scegliere l'opzione "Apri" nel menu File (usando il mouse o premendo prima Alt+F e poi Alt+O).

## Disegnare linee

Conosciamo già il comando **Linea**. L'abbiamo usata per disegnare rettangoli, ma abbiamo anche già anticipato che può essere e più semplicemente usata per disegnare linee, come già si intuisce dal suo stesso nome. Il comando **Linea** senza il parametro R, tratterà una linea larga un pixel, del colore corrente (o di altro colore, se specificato come parametro del comando stesso), da un punto a un altro punto. Il Programma 12 mostra un esempio di come usare l'istruzione **Linea** per tracciare linee.

```
1 ' triangolo.bas
2 ' disegna un triangolo
3 Schermo 12
4 Colore 0, 15
5 Psc
6 Linea (150, 100)-(100, 200)
7 Linea (100, 200)-(200, 200)
8 Linea (200, 200)-(150, 100)
```

Programma 12: Disegna un triangolo

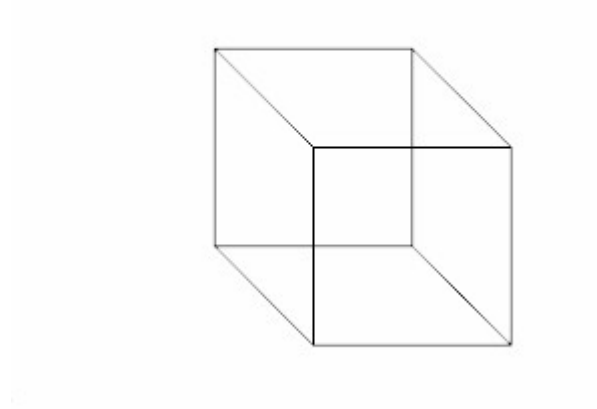


Esempio di esecuzione del programma 12: disegna un triangolo

Il programma seguente è un esempio di cosa si può fare con molte linee. Disegna un cubo sullo schermo.

```
1 ' cubo.bas
2 ' disegna un cubo
3
4 Schermo 12
5 Colore 0, 15
6 Psc
7 ' disegna il quadrato di sotto
8 Linea (150, 150)-(150, 250)
9 Linea (150, 250)-(250, 250)
10 Linea (250, 250)-(250, 150)
11 Linea (250, 150)-(150, 150)
12 ' disegna il quadrato di sopra
13 Linea (100, 100)-(100, 200)
14 Linea (100, 200)-(200, 200)
15 Linea (200, 200)-(200, 100)
16 Linea (200, 100)-(100, 100)
17 ' collega gli angoli
18 Linea (100, 100)-(150, 150)
19 Linea (100, 200)-(150, 250)
20 Linea (200, 200)-(250, 250)
21 Linea (200, 100)-(250, 150)
```

Programma 13: Disegna un cubo



Esempio di esecuzione del programma 13: disegna un cubo

## Disegnare singoli punti sullo schermo

L'ultima istruzione grafica trattata in questo capitolo è **Pixel**. L'istruzione **Pixel** imposta un singolo pixel (punto) sullo schermo.

```


1 ' punti.bas
2 ' usa Pixel per disegnare punti
3 Schermo 12
4 ' imposta il bianco come colore di sfondo
5 Colore , 15
6 ' pulisce lo schermo
7 Psc
8
9 Colore 12 ' rosso
10 Pixel (120, 120)
11 Colore _RGB(255, 102, 0) ' arancione
12 Pixel (137, 137)
13 Colore 14 ' giallo
14 Pixel (149, 149)
15 Colore 10 ' verde
16 Pixel (155, 155)
17 Colore 9 ' blu
18 Pixel (159, 159)
19 Colore _RGB(255, 0, 255) ' magenta
20 Pixel (163, 163)
21 Colore 0 ' nero
22 Pixel (166, 166)

```



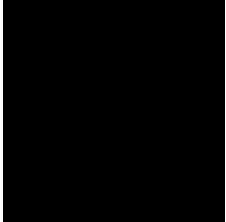
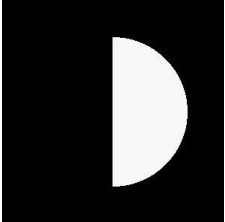
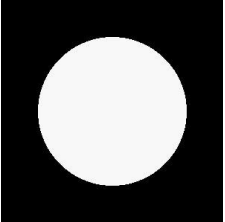
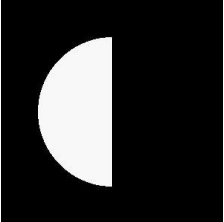
Programma 15: Utilizzare Pixel per disegnare punti



Esempio di esecuzione del programma 15: usare Pixel per disegnare punti

	Pixel (x , y)
 <p><b>Nuovo concetto</b></p>	<p>Il comando <b>Pixel</b> disegna un punto sullo schermo con il colore per disegnare impostato con il comando <b>Colore</b> (preimpostato è il colore bianco, essendo preimpostato il nero come colore di sfondo).</p>

## Esercizi

 <p><b>Cerca le parole</b></p>	<pre> a a t a n i d r o o c r a g g i o l e r u a o i h c r e c t s s z s c k v c e t a u y z a i j l n t c a t e e l a g t a i d n w r h v n r n f a g g q o g a o g a r a e n i l r p s r e e t s l l o a n g n o t a a o g c l p s c p a l t e z z a </pre> <p>Psc, centro, cerchio, colore, coordinata, ciano, grafica, altezza, linea, grafica, raggio, nota, salva, larghezza</p>
 <p><b>Problemi</b></p>	<p>Scrivi un programma che disegni ciascuno dei quarti della luna (luna nuova, primo quarto, luna piena e terzo quarto) e dica il nome del quarto. Suggerimento: disegna la luna come un cerchio e poi disegna un rettangolo sulla parte di luna da non mostrare.</p> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">  <p>Luna nuova</p> </div> <div style="text-align: center;">  <p>Primo quarto</p> </div> <div style="text-align: center;">  <p>Luna piena</p> </div> <div style="text-align: center;">  <p>Terzo quarto</p> </div> </div>