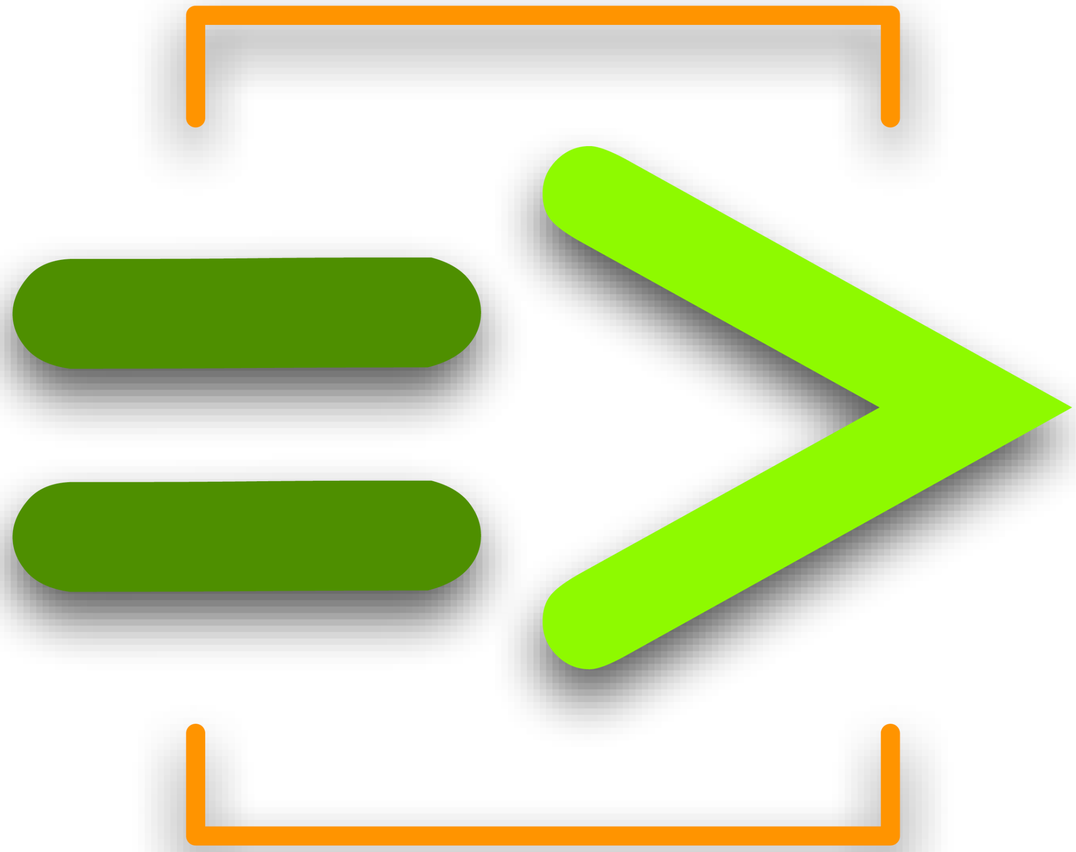
PROGRAMMARE MUSICA AL VOLO CON CHUCK



Chuck é un linguaggio di programmazione per la sintesi audio e la creazione musicale in tempo reale.
ChucK offre un modello di programmazione concorrente basato sul concetto di tempo (Chronos) che é preciso ed espressivo e permette di aggiungere e modificare codice al volo (*on-the-fly*).
Supporta i protocolli MIDI e OpenSoundControl (OSC), le periferiche HID, e l'audio multicanale.
Open source e disponibile gratuitamente per MacOS X, Windows e Linux, necessita di minimi requisiti di sistema ed é compatibile con i processori M1 di Apple.

É stato realizzato da Ge Wang durante il suo Ph.D alla Princeton University sotto la guida di Perry R. Cook.

É facile da imparare e divertente

Ge Wang
Associate Professor
Centre for Computer Research in Music and Acoustics (CCRMA)

Perry R. Cook
Professor Emeritus
Princeton University
Computer Science and Department of Music

MiniAudicle é il leggero ambiente di sviluppo integrato (IDE) di ChucK sviluppato da Spencer Salazar e Ge Wang.

*[*ChucK puó essere tradizionalmente utilizzato da command-line]*

Mac OS X

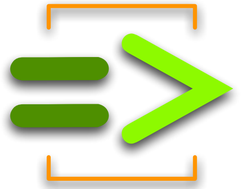
Linux

Windows

- **Dispone di bottoni e comandi per la programmazione al volo**
- **Fornisce un monitor dello status della virtual machine**
- **Un console monitor utile per il debugging**
- **Offre la possibilitá di visualizzare elementi grafici per la user interface (mAUI)**

ROBOT#14

ChucK é un linguaggio fortemente tipizzato, il che vuol dire che i tipi sono determinati al momento della compilazione.

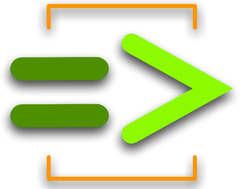


**Tuttavia non é tipizzato staticamente perché il compiler di ChucK é parte della sua macchina virtuale, che é un componente run-time*

Primitivi

- **Float (numeri a virgola mobile a double-precision)**
- **Time**
- **Dur (per durate)**
- **Void (no type)**

Le variabili sono locazioni di memoria che contengono dati. Possiamo assegnare un valore ad una variabile con l'operatore ChucK (=>)

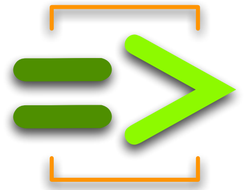


Reference Types

- Object (tipo base da cui ereditano tutte le classi - come in Smalltalk
- Array (insieme N-dimensionale di data dello stesso tipo
- Event (meccanismo di sincronizzazione fondamentale ed estendibile
- UGen (classe base estendibile per gli unit generator
- string (stringa di caratteri)

Complex Types

- Complex (Real + Imaginary)
- Polar (Magnitude + Phase)



Arrays

- **Ci si accede con un numero intero (0 é il primo elemento)**
- **Possono essere usati come mappe associative a cui accedere con stringhe**
- **Sono Oggetti**

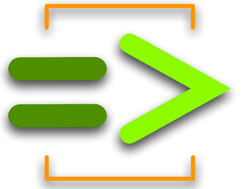
```
int foo [10];
foo.size(); // returns 10

[60, 63, 65, 67] @=> int notes[];
notes.cap(); // returns 4
```

Si possono dichiarare arrays di oggetti, i quali vengono automaticamente insaziati

```
Object group[10]
```

Si possono dichiarare arrays multidimensionali

```
float spalla3d[9][7][12];
```



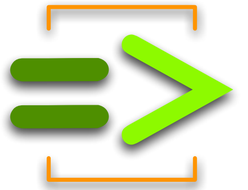
TIME AND TIMING

* Il concetto di tempo é incorporato nel linguaggio

* Il tempo puó solamente avanzare

FUNZIONI

Possiamo scomporre il codice e compiti ricorrenti in unitá individuali



Aumenta la riutilizzabilitá del codice e la sua leggibilitá

```
// definiamo una funzione chiamata ‘play’
fun void play( int arg)
{
    // qui dentro scriviamo cosa fa la funzione
}
```

Questa funzione ha un argomento di tipo *int*

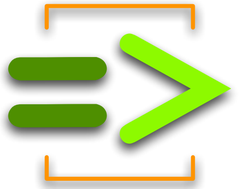
Questa funzione non ritorna alcun valore

- ***Per chiamare una funzione usiamo il nome della funzione e gli argomenti appropriati***

```
play(99) => int result;
```

- ***Le funzioni possono essere sovraccaricate (overloaded) - si possono definire funzioni con lo stesso nome ma con argomenti differenti***

CONCURRENCY AND SHREDS



- ***ChucK puó eseguire molti processi contemporaneamente.***
- ***Un processo in ChucK é chiamato shred.***
- ***Spork a shred significa aggiungere un nuovo processo alla macchina virtuale.***
- **La concorrenza é *sample-synchronous***

```
// definiamo una funzione chiamata ‘techno’
fun void techno( float bpm)
{
   // qui dentro scriviamo cosa fa la funzione
}
```

```
// “sporkiamo la funzione”
spork ~techno (144.5);

// abbiamo bisogno di una funzione genitore
while(true) 1::second now;
```

hex	oct	char	dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char
0	000	NULL	32	20	040	space	64	40	100	@	96	60	140	`
1	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	002	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	004	EOT	36	24	044	$	68	44	104	D	100	64	144	d
5	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	007	BEL	39	27	047	'	71	47	107	G	103	67	147	g
8	010	BS	40	28	050	(	72	48	110	H	104	68	150	h
9	011	TAB	41	29	051	)	73	49	111	I	105	69	151	i
a	012	LF	42	2a	052	*	74	4a	112	J	106	6a	152	j
b	013	VT	43	2b	053	+	75	4b	113	K	107	6b	153	k
c	014	FF	44	2c	054	,	76	4c	114	L	108	6c	154	l
d	015	CR	45	2d	055	-	77	4d	115	M	109	6d	155	m
e	016	SO	46	2e	056	.	78	4e	116	N	110	6e	156	n
f	017	SI	47	2f	057	/	79	4f	117	O	111	6f	157	o
10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	p
11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	y
1a	032	SUB	58	3a	072	:	90	5a	132	Z	122	7a	172	z
1b	033	ESC	59	3b	073	;	91	5b	133	[	123	7b	173	{
1c	034	FS	60	3c	074	<	92	5c	134	\	124	7c	174	|
1d	035	GS	61	3d	075	=	93	5d	135	]	125	7d	175	}
1e	036	RS	62	3e	076	>	94	5e	136	^	126	7e	176	~
1f	037	US	63	3f	077	?	95	5f	137	_	127	7f	177	DEL

www.alpharithms.com

ASCII - abbreviazione di American Standard Code for Interchange é uno standard di codificazione dei caratteri a 8 bit per la comunicazione elettronica

ROBOT#14