10 programmi per la calcolatrice CASIO fx-180P

10/09/2006
Revisione 0.1

Indice

Indovina il numero...

Scopo del gioco è indovinare il numero "pensato" dalla calcolatrice.
Il gioco è realizzato mediante due programmi: il programma P2 inizializza il gioco, generando un numero compreso tra 0 e 999 (estremi inclusi), che rappresenta il numero da indovinare, ed azzerando il contatore dei tentativi effettuati; il programma P1 che risponde -1 se il numero proposto dall'utente è inferiore al numero da indovinare, 0 se uguale, 1 se superiore.

Uso delle memorie

K1..... contiene il numero di tentativi effettuati
K2..... contiene il numero da indovinare

Note implementative

x^2 √..... determina il valore assoluto
`× (Kout N + 1) Kin N ÷ Kout N =`... incrementa la memoria KN, preservando il valore dell'espressione in corso di valutazione

Esempio d'uso

Sequenza di inizializzazione del programma

```
P2 [0]
```

Uso del programma

L'utente prova con 500
`500 P1 [-1]`
500 è inferiore al numero da indovinare, l'utente gioca 750:
`750 P1 [1]`
750 è maggiore del numero segreto, l'utente tenta con 675:
`675 P1 [0]`
L'utente ha indovinato! Il numero di tentativi effettuati è contenuto nella memoria K1:
`Kout 1 [3]`
L'utente ha impiegato 3 tentativi per indovinare il numero segreto.

P2

1	RAN#	genera un numero casuale compreso tra 0.000 e 0.999
2	×	moltiplicandolo per 1000, lo si porta nel campo [0, 999]
3	1	
4	EXP	
5	3	

6	=	
7	Kin 2	il numero “segreto” viene memorizzato in K2
8	0	
9	Kin 1	azzera il contatore dei tentativi (ed elimina il numero “segreto” dal display)

P1

10	-	
11	Kout 2	
12	=	determina la differenza tra il tentativo dell'utente ed il numero segreto
13	×	
14	(	incrementa il contatore dei tentativi mediante la formula
15	Kout 1	(Kout 1 + 1) Kin 1 / Kout 1
16	+	che, pur lasciando inalterato il valore della differenza appena calcolata
17	1	
18	)	
19	Kin 1	
20	÷	
21	Kout 1	
22	=	
23	Min	memorizza temporaneamente il valore della differenza nella memoria M
24	x^2	determina il valore assoluto della differenza
25	√	
26	÷	fornisce l'esito del tentativo come risultato dell'espressione
27	(	|d| / (d + ε)
28	MR	che per d non nullo torna il segno della differenza, 0 altrimenti
29	+	
30	1	
31	EXP	
32	9	
33	9	
34	+/-	
35	)	
36	=	

Pensa un numero...

Scopo del gioco è aiutare la calcolatrice ad indovinare il numero pensato dall'utente, rispondendo “troppo alto”/“troppo basso” alle proposte che di volta in volta la calcolatrice elabora. Il numero segreto può essere scelto a piacere nell'intervallo [0, 1023]

Se l'utente non bara, la calcolatrice riesce sempre a scoprire il numero “segreto” effettuando al più 10 giocate.

Il gioco è realizzato mediante due programmi: il programma P2 inizializza il gioco ed effettua la prima giocata; il programma P1 elabora il prossimo numero da proporre sulla base della risposta dell'utente al turno precedente.

L'utente deve rispondere con -1 se il numero segreto è inferiore a quello proposto dalla calcolatrice, 1 in caso contrario.

Uso delle memorie

K1..... contiene il numero di tentativi effettuati

K2..... contiene l'ultimo numero proposto all'utente

Esempio d'uso

L'utente sceglie 384 come numero “segreto”

Sequenza di inizializzazione del programma

P2 [512]

Uso del programma

L'utente risponde -1 per segnalare che il numero “segreto” è inferiore a quello proposto:

```
1 +/- P1 [256]
```

L'utente risponde 1 per segnalare che il numero “segreto” è inferiore a quello proposto:

```
1 P1 [384]
```

La calcolatrice ha già indovinato; per verificare il numero di giocate effettuate dalla calcolatrice si ispeziona il contenuto della memoria K1:

```
Kout 1 [3]
```

La calcolatrice ha effettuato 3 tentativi prima di scoprire il numero “segreto”.

P2

1	`1`	inizializza il contatore di tentativi effettuati
2	`Kin 1`	
3	`5`	imposta il primo tentativo da effettuare
4	`1`	si tratta del centro dell'intervallo di gioco [0, 1023]
5	`2`	il gioco proseguirà bisezionando l'intervallo contenente
6	`Kin 2`	il numero “segreto”, seguendo le indicazioni fornite dall'utente

P1

7	×	incrementa il contatore di tentativi (cfr. “Indovina il numero segreto”)
8	(	
9	Kout 1	
10	+	
11	1	
12	)	
13	Kin 1	
14	÷	
15	Kout 1	
16	=	
17	×	biseziona l'intervallo corrente, giocando il numero centrale del semi-intervallo
18	2	inferiore se l'utente ha risposto -1 (numero “segreto” inferiore a quello
19	x^y	precedentemente proposto), quello superiore se l'utente ha risposto +1 (numero
20	(	“segreto” superiore a quello proposto)
21	1	
22	0	trattandosi di una ricerca sul campo [0, 1023], si fa uso di una semplice
23	-	bisezione binaria, seguendo lo schema:
24	Kout 1	
25	)	$t_{n+1} = t_n + r \times 2^{10-n}$
26	=	
27	+	ove t_n rappresenta l'n-esimo numero proposto, r è la risposta dell'utente
28	Kout 2	
29	=	
30	Kin 2	

Lancio di due dadi

Il programma simula il lancio di due dadi.

Nota: disattivare l'uso dei decimali (FIX 0) prima di giocare.

Note implementative

`RND`..... rende effettivo l'arrotondamento in atto;
quando opera in modalità FIX, la calcolatrice visualizza il dato corrente con il numero di cifre decimali richieste, ma utilizza tutte quelle a sua disposizione per la valutazione dell'espressione; l'istruzione RND rende il dato interno uguale a quello visualizzato

Esempio d'uso

Uso del programma

```
P1 [41]
P1 [35]
P1 [62]
```

P1

1	`RAN#`	genera un numero casuale in [0.000, 0.999]
2	`×`	moltiplica il numero per 6.006, coprendo il campo [0.000, 5.994]
3	`6`	
4	`.`	
5	`0`	
6	`0`	
7	`6`	
8	`+`	somma 0.5, coprendo il campo [0.500, 6.499]
9	`.`	
10	`5`	
11	`=`	
12	`RND`	effettua l'arrotondamento in [1, 6]
13	`×`	moltiplica per 10, per “far posto” al secondo dado...
14	`1`	
15	`0`	
16	`+`	

17	(	determina il punteggio del secondo dado...
18	RAN#	
19	×	
20	6	
21	.	
22	0	
23	0	
24	6	
25	+	
26	.	
27	5	
28	)	
29	RND	
30	=	mostra i punteggi dei due dadi “affiancati”

Caccia al tesoro

Scopo del gioco è scoprire dove è stato sepolto il tesoro.

Il gioco è realizzato mediante due programmi: il programma P2 inizializza il gioco, richiedendo la dimensione del campo di gioco (legato alla difficoltà della ricerca da compiere), generando due numeri casuali che rappresentano le coordinate da indovinare, ed azzerando il contatore dei tentativi effettuati; il programma P1 che dà un'indicazione di quanto vicino al tesoro si trovi il punto identificato dalle coordinate immesse dall'utente. Il gioco termina quando l'utente scopre l'ubicazione del tesoro, evento segnalato dalla calcolatrice emettendo uno zero.

Nota: disattivare l'uso dei decimali (FIX 0) prima di giocare.

Uso delle memorie

K1..... contiene il numero di tentativi effettuati
K2..... contiene la coordinata x del punto da indovinare
K3..... contiene la coordinata y del punto da indovinare
K4..... dimensione del campo di gioco

Note implementative

`X↔K N + 1 = X↔K N`..... incrementa la memoria KN, preservando il valore dell'espressione in corso di valutazione

Esempio d'uso

Sequenza di inizializzazione del programma

Inizializza un campo di gioco di 5x5 caselle:

```
4 P2 [0]
```

Uso del programma

```
P1
```

L'utente tenta la posizione (4,0):

```
4 RUN 0 RUN [2]
```

La distanza dichiarata è 2; l'utente prova con (2,0):

```
2 RUN 0 RUN [2]
```

La distanza rimane 2; l'utente prova a spostarsi lungo l'asse y, con (2,1):

```
2 RUN 1 RUN [1]
```

La distanza è scesa a 1; si prosegue lungo la stessa direzione, provando (2,2):

```
2 RUN 2 RUN [0]
```

Il tesoro è stato raggiunto!

```
Kout 1 [4]
```

L'ubicazione del tesoro è stata scoperta dopo 4 tentativi.

P2

1	`ENT`	riceve le dimensioni del campo di gioco (intero nell'intervallo [1, 9])
2	`Kin 4`	memorizza le dimensioni del campo di gioco nella memoria K4
3	`RAN#`	genera la coordinata x nel campo [0, 9]
4	`×`	
5	`Kout 4`	
6	`=`	
7	`RND`	
8	`Kin 2`	memorizza la coordinata x del punto da indovinare nella memoria K2
9	`RAN#`	genera la coordinata y nel campo [0, 9]
10	`×`	
11	`Kout 4`	
12	`=`	
13	`RND`	
14	`Kin 3`	memorizza la coordinata y del punto da indovinare nella memoria K3
15	`0`	azzera il contatore dei tentativi
16	`Kin 1`	

P1

17	`ENT`	riceve la coordinata x del punto pensato dall'utente
18	`-`	determina la distanza dalla coordinata x del tesoro: |ENT - K2|
19	`Kout 2`	
20	`=`	
21	x^2	
22	`X↔K 1`	incrementa il contatore dei tentativi
23	`+`	
24	`1`	
25	`=`	
26	`X↔K 1`	richiama la distanza precedentemente calcolata sull'asse delle ascisse
27	`+`	
28	`(`	
29	`ENT`	determina la distanza dalla coordinata y del tesoro: |ENT - K3|
30	`-`	
31	`Kout 3`	

32	)	
33	x^2	
34	=	
35	√	determina la distanza euclidea
36	RTN	riesegue da capo il programma

Scegli un numero

Scopo del gioco è dimostrare che la calcolatrice è in grado di indovinare il numero scelto a piacere dall'utente nell'intervallo [1, 15], se questi indica correttamente in quali colonne del seguente specchietto esso si trova:

15	15	15	15
13	14	14	14
11	11	13	13
9	10	12	12
7	7	7	11
5	6	6	10
3	3	5	9
1	2	4	8
1	**2**	**3**	**4**

Il gioco è realizzato tramite un programma che propone in successione all'utente i numeri 1, 2, 3 e 4, ed attende la risposta dell'utente che deve essere 1 se il numero è presente nella colonna omonima, 0 se è assente.

La risoluzione del problema è banale quando si nota che la disposizione è tale per cui ogni numero è inserito nell'i-esima colonna se e solo se l'i-esimo bit della sua rappresentazione binaria è 1 (considerando il bit meno significativo in posizione 1 ed il bit più significativo in posizione 4).

Il programma pertanto si limita a calcolare il numero decimale rappresentato dai “bit” forniti dall'utente come risposte alle 4 domande poste.

Uso delle memorie

M..... rappresentazione binaria “in divenire” del numero scelto dall'utente

Esempio d'uso

Uso del programma

L'utente sceglie il numero 11, quindi avvia il programma:

```
P1 [1]
```

poiché il numero 11 compare in colonna 1, l'utente risponde 1:

```
1 RUN [2]
```

poiché il numero 11 compare anche in colonna 2, l'utente risponde 1:

```
1 RUN [3]
```

il numero 11 non compare in colonna 3, l'utente perciò risponde 0:

```
0 RUN [4]
```

il numero 11 compare in colonna 4, quindi l'utente risponde 1:

```
1 RUN [11]
```

La calcolatrice ha indovinato!

P1

1	0	
2	Min	azzera l'accumulatore
3	1	visualizza sul display il numero 1
4	×	
5	ENT	attende la risposta circa la presenza del numero in colonna 1
6	M+	aggiorna l'accumulatore
7	2	visualizza sul display il numero 2
8	×	
9	ENT	attende la risposta circa la presenza del numero in colonna 2
10	M+	aggiorna l'accumulatore
11	3	visualizza sul display il numero 3
12	×	
13	ENT	attende la risposta circa la presenza del numero in colonna 3
14	×	il bit in posizione 3 pesa 4, quindi, per procedere correttamente,
15	4	si rende necessario moltiplicare per 4 e dividere per 3...
16	÷	
17	3	
18	M+	aggiorna l'accumulatore
19	4	visualizza sul display il numero 4
20	×	
21	ENT	attende la risposta circa la presenza del numero in colonna 4
22	×	il bit in posizione 4 pesa 8, quindi, per procedere correttamente,
23	2	si rende necessario moltiplicare ulteriormente per 2...
24	M+	aggiorna l'accumulatore
25	MR	visualizza il numero decimale risultante

Morra cinese

Il programma realizza il classico gioco della morra cinese, utilizzando la codifica:

SASSO → 1

CARTA → 2

FORBICE → 3

La mappa seguente ricorda lo schema di gioco, riportando, per ogni giocata, il vincitore della partita:

Giocatore A				
	F	**B**	**A**	**-**
	C	**A**	**-**	**B**
	S	**-**	**B**	**A**
		S	**C**	**F**
		Giocatore B		

Supponendo di assegnare 1 punto per la vittoria, 0 per il pareggio e -1 in caso di sconfitta, la situazione di gioco dal punto di vista del giocatore B diventa la seguente:

Calcolatrice				
	3	**+1**	**-1**	**0**
	2	**-1**	**0**	**+1**
	1	**0**	**+1**	**-1**
		1	**2**	**3**
		Utente		

Osservando che le diagonali della matrice ospitano valori uguali, si desume che il punteggio dell'utente dipende esclusivamente dalla differenza delle due giocate. Più precisamente, detta y la giocata della calcolatrice e x quella dell'utente, posto d=x-y, si ottiene che il punteggio p ottenuto dall'utente in una generica partita è dato da:

p = +1 se d = -2, 1

p = 0 se d = 0

p = -1 se d = -1, 2

Tale risultato si può ottenere utilizzando un polinomio di 5 grado, che si rivelerebbe tuttavia intrattabile per un programma che può essere lungo al più 38 passi. Si opta per una soluzione alternativa, ma equivalente, ovvero una sinusoide:

p = round(sin ⅔πd)

Nota: impostare l'uso dei gradi sessagesimali (DEG) e disattivare l'uso dei decimali (FIX 0) prima di iniziare il gioco. Prima di ogni ciclo di partite, inoltre, assicurarsi di azzerare le memorie (KAC).

Uso delle memorie

M..... esito dell'ultima partita disputata
K1..... contiene il numero di partite giocate
K2..... contiene il numero di partite vinte

Esempio d'uso

Uso del programma

`P1 [0]`
L'utente gioca SASSO:
`1 RUN [2] RUN [-1]`
la calcolatrice ha giocato CARTA, e vince; l'utente gioca FORBICE:
`3 RUN [3] RUN [0]`
La calcolatrice ha giocato FORBICE, evidenziando un pareggio; l'utente gioca CARTA:
`2 RUN [3] [-1]`
la calcolatrice ha giocato FORBICE, e vince; l'utente gioca nuovamente CARTA:
`2 RUN [1] [1]`
La calcolatrice ha giocato SASSO, e perde.
Per verificare lo stato delle partite giocate e vinte, si esaminano i contenuti delle memorie K1 e K2:
`Kout 1 [4] Kout 2 [1]`
Sono state giocate 4 partite, delle quali una solamente vinta dall'utente.

P1

1	ENT	attende la giocata dell'utente
2	-	
3	(	genera la giocata della calcolatrice
4	RAN#	genera un numero casuale nel campo [0.000, 0.999]
5	×	
6	3	moltiplica il numero per 3, coprendo il campo [0.000, 2.997]
7	+	
8	.	
9	5	somma 0.5, coprendo il campo [0.500, 3.497]
10	)	
11	RND	effettua l'arrotondamento in [1, 3]
12	HLT	visualizza la giocata della calcolatrice
13	=	determina la differenza d tra le due giocate
14	×	determina il punteggio per l'utente
15	1	
16	2	

17	`0`	
18	`=`	
19	`SIN`	
20	`RND`	P=RND(SIN(d×120))
21	`Min`	memorizza temporaneamente il risultato nella memoria M
22	x^2	determina l'incremento del numero di partite vinte dall'utente:
23	`+`	1 se P = +1; 0 se P = 0, -1
24	`MR`	la formula implementata è $(P + P^2) / 2$
25	`=`	
26	`÷`	
27	`2`	
28	`+`	
29	`Kout 2`	aggiungi l'incremento al conteggio attuale delle partite vinte dall'utente
30	`=`	
31	`Kin 2`	
32	`X↔K 1`	incrementa di 1 il contatore delle partite giocate
33	`+`	
34	`1`	
35	`=`	
36	`X↔K 1`	
37	`MR`	visualizza il punteggio dell'ultima partita
38	`RTN`	predisponiti per una nuova partita

Ritorno al futuro

Scopo del gioco è simulare un viaggio interstellare svolto ad una velocità prossima a quella della luce. Accade così che il tempo sulla navicella scorre più lentamente rispetto a quello terrestre.

Il gioco è realizzato mediante un programma che richiede il tempo terrestre che deve trascorrere tra il decollo e l'atterraggio della navicella spaziale (i valori tipici sono compresi tra 100 e 200 anni). Successivamente richiede la lunghezza del viaggio che si intende intraprendere, espresso in anni-luce, quindi la velocità di crociera, espressa come frazione della velocità della luce (i valori tipici sono compresi tra 0,75 e 0,99). Il programma determina la durata del viaggio così come misurato dall'orologio della navicella, producendo una condizione d'errore nel caso questi superi i 70 anni[1]; successivamente visualizza il tempo trascorso sulla terra, emettendo infine uno 0 se questi differisce al più 5 anni dal tempo inizialmente programmato, con condizione d'errore in caso contrario.

Nota: per comodità conviene limitare il numero di cifre decimali in uso (FIX 2/FIX 3) durante il gioco.

Fisica (approssimata) del gioco

Poiché la navicella percorre la distanza d a velocità costante v = kc, il tempo impiegato è:

$$t_{NAV} = d / k$$

poiché la navicella procede ad una velocità prossima a quella della luce, il tempo trascorso sulla terra è invece:

$$t_{TERRA} = t_{NAV} / \sqrt{(1 - k^2)}$$

Uso delle memorie

K1..... durata programmata del viaggio, espressa in anni “terrestri”
K2..... distanza da percorrere durante il viaggio, espressa in anni-luce
K3..... velocità di crociera, espressa come frazione della velocità della luce
K4..... durata del viaggio, espressi in anni “navicella”

Esempio d'uso

Uso del programma

Si vuole effettuare un viaggio della durata di 100 anni:

```
P1 [0] 100
```

il viaggio prevede di viaggiare per 30 anni luce, alla velocità di 0,90 c:

```
RUN [100.00] 30 RUN [30.00] .9
```

il tempo trascorso all'interno della navicella è pari a (tutt i tempi sono espressi in anni):

```
RUN [33.33]
```

mentre sulla terra è trascorso il tempo:

```
RUN [76.47]
```

1 nell'ipotesi che l'aspettativa media dell'utente si aggiri sui cent'anni!

il viaggio si è concluso in anticipo rispetto ai 100 anni programmati:
`RUN [E]`
l'utente prova ad aumentare la distanza da percorrere:
`P1 [0] 100 RUN [100.00] 70 RUN [70.00] .9 RUN [77.78] RUN [E]`
il viaggio è durato più di 77 anni, troppi anche per un giovane astronauta!
L'utente prova ad aumentare la velocità di crociera:
`P1 [0] 100 RUN [100.00] 30 RUN [30] .95 RUN [31.58] RUN [101.13] RUN [0]`
l'atterraggio è avvenuto con un solo anno di ritardo: missione compiuta!

P1

1	ENT	attende l'immissione della durata programmata del viaggio
2	Kin 1	
3	ENT	attende l'immissione della distanza da percorrere
4	Kin 2	
5	÷	
6	ENT .5	attende l'immissione della velocità di crociera
7	Kin 3	(il dato 0,5 serve ad evitare una condizione d'errore che si verificherebbe più avanti)
8	=	
9	Kin 4	
10	HLT	visualizza il tempo trascorso sulla navicella
11	+/-	verifica che il tempo trascorso sulla navicella sia inferiore a 70 anni
12	+	
13	7	
14	0	
15	=	
16	√	
17	Kout 4	
18	÷	
19	(	
20	1	
21	-	
22	Kout 3	
23	x^2	
24	)	
25	√	
26	=	
27	HLT	visualizza il tempo trascorso sulla terra
28	-	valuta la differenza tra la durata effettiva del viaggio e quella programmata
29	Kout 1	
30	=	

31	x^2	verifica che la discrepanza sia inferiore a 5 anni
32	√	
33	+/-	
34	+	
35	5	
36	=	
37	√	
38	0	visualizza uno zero per indicare il successo della missione

Gioco del 21

Scopo del gioco è estrarre una dopo l'altra una sequenza di carte da gioco, avvicinandosi il più possibile al punteggio di 21 senza superarlo. Il punteggio è dato dalla somma dei valori numerici delle singole carte estratte, attribuendo il valore 10 alle figure, 1 all'asso.

Il gioco è realizzato mediante due programmi: il programma P2 inizializza il gioco, mentre il programma P1 effettua l'estrazione delle carte in successione.

Per ovvi motivi legati alle limitazioni della calcolatrice, il gioco implementato prevede il reinserimento della carta estratta nel mazzo.

Nota: disattivare l'uso dei decimali (FIX 0) prima di giocare.

Uso delle memorie

M..... 21 (limite oltre il quale la partita si considera persa)

K1..... contiene il punteggio corrente

Note implementative

`0 1/x AC`..... genera una condizione d'errore;
applicabile solo in coda al programma

Esempio d'uso

Sequenza di inizializzazione del programma

```
P2 [0]
```

Uso del programma

```
P1 [0]
```

Al momento, il punteggio dell'utente è nullo;

```
RUN [6] RUN [6]
```

la carta estratta è un 6; il punteggio totale è 6; l'utente decide di proseguire:

```
RUN [1] RUN [7]
```

è stato estratto un asso, per cui il punteggio sale a 7; l'utente decide di andare avanti:

```
RUN [10] RUN [17]
```

è uscito un 10; il punteggio totale diventa 17; l'utente decide di chiamare un'altra carta:

```
RUN [5] RUN [E]
```

è stato estratto un 5, che fa oltrepassare il punteggio limite di 21: errore!

P2

1	`2`	inizializza il punteggio massimo
2	`1`	
3	`Min`	
4	`0`	inizializza il punteggio corrente

5	`Kin`	

P1

6	`Kout 1`	visualizza il punteggio corrente
7	`HLT`	
8	`RAN#`	genera un numero casuale nel campo [0.000, 0.999]
9	`×`	moltiplica il numero per 10.01, coprendo il campo [0.000, 9.999]
10	`1`	
11	`0`	
12	`.`	
13	`0`	
14	`1`	
15	`+`	
16	`.`	
17	`5`	somma 0.5, coprendo il campo [0.500, 10.499]
18	`=`	
19	`RND`	effettua l'arrotondamento in [1, 11]
20	`HLT`	visualizza il valore numerico della carta estratta
21	`+`	aggiorna il punteggio dell'utente
22	`Kout 1`	
23	`=`	
24	`Kin 1`	
25	`x≤M`	se il punteggio è minore del valore in memoria M, il gioco continua...
26	`0`	in caso contrario, si genera una situazione d'errore (divisione per zero)
27	`1/x`	
28	`AC`	

Balistica

Scopo del gioco è colpire un bersaglio fisso, posto ad una distanza inferiore alla gittata, regolando l'alzo del cannone a disposizione.
Il gioco è realizzato mediante due programmi: il programma P2 inizializza il gioco, mentre il programma P1 effettua il tiro, determinandone l'esito.

Nota: per comodità conviene utilizzare una sola cifra decimale (FIX 1) durante il gioco.

Fisica (approssimata) del gioco

Per facilitare il gioco, l'unica variabile indipendente è l'alzo α. La variabile dipendente cercata è la distanza cui il proiettile tocca terra. Detta h(t) l'altezza del proiettile e v il modulo della velocità, si determina il tempo di caduta tramite la relazione:

$$h(t) = v \sin \alpha \, t - \tfrac{1}{2} g t^2$$

dalla quale si ricava, ponendo h(t) = 0:

$$t = 2 v \sin \alpha / g$$

lo spazio percorso dal proiettile durante il volo è perciò:

$$s = v \cos \alpha \, t$$

Sostituendo il valore di t precedentemente ricavato si ottiene:

$$s = 2 v^2 \cos \alpha \sin \alpha / g$$

ovvero:

$$s = v^2 \sin 2\alpha / g$$

Derivando in α e annullando si ottiene che la gittata massima si ha per $\alpha = 45°$, e vale:

$$s_{max} = v^2 / g$$

Ponendo per semplicità v = 100 m/s e g = 10 m/s², la gittata diventa pari a 1 Km.

Uso delle memorie

M..... numero di colpi effettuati
K1..... distanza del bersaglio dalla postazione di tiro
K2..... tolleranza di tiro (distanza entro la quale il colpo viene assegnato)

Esempio d'uso

Sequenza di inizializzazione del programma

```
P2 [0]
```

Uso del programma

```
P1 [560.0]
```

Prima del tiro, il programma visualizza la posizione del bersaglio; si prova un alzo di 30°:

```
30 RUN [866.0] RUN [560.0]
```

il tiro è troppo lungo, l'alzo va abbassato:

```
20 RUN [642.8] RUN [560.0]
```
il tiro è ancora lungo, l'alzo va abbassato ulteriormente:
```
15 RUN [500.0] RUN [560.0]
```
tiro corto; l'alzo va leggermente aumentato:
```
17 RUN [559.2] RUN [0]
```
Il bersaglio è stato colpito! Il numero di tiri effettuati (5) visibile richiamando la memoria M:
```
MR [5.0]
```

P2

1	RAN#	determina la posizione del bersaglio nel campo [0.0, 999.0]
2	×	
3	1	
4	EXP	
5	3	
6	=	
7	Kin 1	memorizza la posizione del bersaglio nella memoria K1
8	1	
9	0	
10	Kin 2	K2 = 10 (impostazione della tolleranza di tiro)
11	0	
12	Min	azzera il contatore dei tiri effettuati

P1

13	1	aggiorna il contatore dei tiri effettuati
14	M+	
15	Kout 1	visualizza la distanza cui si trova il bersaglio
16	ENT	attendi l'immissione dell'alzo
17	×	determina la posizione di caduta del colpo
18	2	
19	=	
20	sin	
21	×	
22	1	
23	EXP	
24	3	
25	=	
26	HLT	visualizza il punto di caduta del colpo
27	-	determina la distanza dal bersaglio
28	Kout 1	
29	=	

30	x^2	
31	√	
32	-	
33	Kout 2	
34	=	
35	x>0	se la distanza è superiore alla tolleranza di tiro, si spara un nuovo colpo,
36	0	in caso contrario si segnala il centro visualizzando uno zero

Modulo lunare

Scopo del gioco è riuscire a far toccare al modulo di allunaggio la superficie lunare ad una velocità consona, provvedendo ad azionare i motori quando opportuno, evitando nel contempo di esaurire il carburante a disposizione.

Il gioco è realizzato mediante un programma che, passo dopo passo, indica l'altitudine alla quale si trova il modulo di allunaggio, la sua velocità verticale corrente e la quantità di carburante disponibile nei serbatoi; l'utente, valutando i dati forniti dagli "strumenti di bordo", decide se è opportuno accendere i motori, e quanto carburante consumare durante il passo corrente. In funzione del carburante consumato, la velocità di discesa del modulo risulterà ridotta nella misura di 1 m/s per ogni kg di carburante utilizzato. L'allunaggio, affinché possa essere considerato valido, deve avvenire con una velocità verticale finale inferiore a 2 m/s; in caso contrario il modulo di allunaggio sarà considerato irrimediabilmente distrutto.

Nota: introdurre la sequenza `KAC 5 Kin 3` *prima di iniziare la programmazione.*

Fisica (approssimata) del gioco

Le variabili considerate sono la quota q, la velocità v, la quantità di carburante c e la quantità di carburante cc utilizzata per ottenere la spinta s.

Sono prefissati i valori delle variabili dipendenti iniziali q(0), v(0), c(0).

Al passo p, l'utente può decidere di utilizzare una quantità di carburante pari a:

$$cc(p) \leq c(p)$$

la quantità di carburante consumato determina la spinta risultante sul modulo lunare, che per semplicità si considera numericamente equivalente al carburante utilizzato:

$$s(p) = cc(p)$$

la velocità verticale del modulo, considerando l'accelerazione di gravità e l'eventuale spinta dovuta all'accensione del motore, è espressa dall'equazione ($\Delta t = 1$):

$$v(p+1) = v(p) - g + s(p)$$

e di conseguenza, la quota alla quale si troverà il modulo al passo p+1 sarà:

$$q(p+1) = q(p) + v(p) + 0.5 * a(t)$$

essendo $a(p) = (v(p+1) - v(p))$, per sostituzione si ottiene:

$$q(p+1) = q(p) + 0.5 * (v(p+1) + v(p))$$

detto $p_a = \min_p \{ q(p) \leq 0 \}$, l'allunaggio si considererà riuscito se $|v(p_a)| < 2$.

Uso delle memorie

M..... quantità di carburante consumata cc(p), successivamente v(p+1)

K1..... quota corrente q(p)

K2..... quantità di carburante disponibile c(p)

K3..... velocità verticale corrente v(p)

Note implementative

`M - Kout N x² = √` ..... verifica la condizione |KN| < √M,
generando un errore in caso di esito negativo

Esempi d'uso

Sequenza di inizializzazione del programma

E' necessario impostare i valori q(0), v(0), c(0) nelle rispettive memorie.

```
100 Kin 1 50 Kin2 10 +/- Kin3
```

Uso del programma

```
P1
[100] RUN [-10] RUN [50]
```

Il programma mostra in sequenza quota, velocità, carburante disponibile;

risultano 50kg di carburante disponibili; l'utente prova a bruciarne 60:

```
60 RUN [100] RUN  [-10] RUN [50]
```

il programma rifiuta di consumare più carburante di quello disponibile, e ripete la richiesta; l'utente opta in questo caso per una caduta libera, lasciando il motore sempre spento: la velocità di discesa aumenta incontrollata fino all'inevitabile schianto finale:

```
0 RUN [87.5] RUN [-15] RUN [50]
0 RUN [70] RUN [-20] RUN [50]
0 RUN [47.5] RUN [-25] RUN [50]
0 RUN [20] RUN [-30] RUN [50]
0 RUN [E]
```

L'utente re-inizializza gli strumenti di bordo:

```
100 Kin 1 50 Kin2 10 +/- Kin3
```

e riavvia il programma di allunaggio:

```
P1
```

in questo caso l'utente effettua un allunaggio perfetto (velocità finale nulla):

```
[100] RUN [-10] RUN [50]
0 RUN [87.5] RUN [-15] RUN [50]
0 RUN [70] RUN [-20] RUN [50]
0 RUN [47.5] RUN [-25] RUN [50]
15 RUN [27.5] RUN [-15] RUN [35]
10 RUN [15] RUN [-10] RUN [25]
10 RUN [7.5] RUN [-5] RUN [15]
5 RUN [2.5] RUN [-5] RUN [10]
10 RUN [0]
```

al termine del programma, Kout 2 contiene la quantità di carburante residuo, Kout 3 la velocità finale:

```
Kout 2 [0] Kout 3 [0]
```

P1

1	`Kout 1`	visualizza q(p)
2	`HLT`	
3	`Kout 3`	visualizza v(p)

4	HLT	
5	Kout 2	visualizza c(p)
6	ENT	richiede il valore di cc(p)
7	Min	
8	-	
9	Kout 2	verifica che cc(p) ≤ c(p)
10	=	
11	x>0	se cc(p) > c(p), riesegui la procedura di richiesta carburante
12	+/-	aggiorna c(p+1)
13	Kin 2	
14	(	determina v(p+1) utilizzando la memoria M come area di lavoro
15	5	assume g = 5
16	M-	
17	Kout 3	
18	M+	a questo punto, la memoria M contiene v(p+1)
19	+	
20	MR	
21	Kin 3	v(p+1) viene salvata nella memoria K3
22	)	
23	÷	
24	2	
25	+	sul display è ora presente il valore 0.5 * (v(p) + v(p+1))
26	Kout 1	che sommato al valore q(p)
27	=	
28	Kin 1	origina q(p+1), che viene immediatamente salvato nella memoria K1
29	x>0	se q(p+1) è positivo, la discesa non è terminata: si effettua un altro passo
30	4	il modulo ha toccato “terra”!
31	-	la verifica |v(p+1)| ≤ 2 avviene valutando l'espressione:
32	Kout 3	$\sqrt{(4 - v(p+1)^2)}$
33	x^2	che esiste solo per i valori ammissibili di v(p+1)
34	=	
35	√	se v(p+1) è troppo elevata, si produce una condizione d'errore,
36	0	in caso contrario segnala si l'avvenuto allunaggio visualizzando quota 0

Valutazione delle prestazioni

Il programma effettua un conteggio alla rovescia a partire dal numero visualizzato sul display, terminando quando il contatore si azzera.

Note implementative

`x>0`..... effettua un salto condizionato all'inizio
del programma se il valore sul display è non nullo

Esempio d'uso

Uso del programma

Effettuare un conteggio da 1000 cicli:
`1000 P1 [0]`
La calcolatrice impiega circa 130 secondi per completare il conto alla rovescia.

P1

1	-	decrementa il numero visualizzato sul display
2	1	
3	=	
4	x>0	se il contatore non è giunto a zero, ricomincia dal passo 1