# Informazioni sul generatore

Si premette subito che un computer, in quanto macchina assolutamente priva di arbitrio (ovvero deterministica), non è in grado di generare dei numeri casuali nel senso letterale del termine.
Quello che si può fare è realizzare un algoritmo che generi una sequenza di numeri avente le stesse proprietà statistiche di una sequenza di numeri casuali.
Agli effetti pratici la sequenza generata ha lo stesso valore di una sequenza realmente casuale, ma il fatto che si utilizzi un algoritmo implica che è possibile ricreare la stessa sequenza fornendo all'algoritmo lo stesso valore iniziale (chiamato appunto *seme*).
Pertanto è più corretto parlare di generatori di numeri *pseudo*-casuali anziché di numeri casuali veri e propri.

L'algoritmo utilizzato in questo sito è noto in letteratura e la sua "bontà", nel senso che genera una sequenza con le stesse proprietà statistiche di una sequenza casuale, è stata dimostrata, con i test statistici del caso.
Si tratta di un generatore di Lehmer, ovverosia un generatore *congruenziale moltiplicativo*.
L'algoritmo, a partire dal seme identificato con $X_0$, è definito in termini ricorsivi nel seguente modo:

$$X_{k+1} = (aX_k+c) \bmod m, \mathrm{k} > 0$$

I numeri generati appartengono all'intervallo [0, *m-1*] e

*m* è un numero intero
*a* e *c* sono numeri interi maggiori o uguali a 0 e minori di *m*

Di particolare importanza per la bontà della sequenza è la scelta dei valori di *a*, *c* e *m*. Nel caso di questo sito i valori scelti sono stati tratti dalla letteratura e sono: *m*=**2147483647** ($2^{31}$-1), *a*=**1103515245** e *c*=**0**.

L'implementazione deve gestire i casi di *overflow* che possono capitare

durante la moltiplicazione $aX_k$. Dato che dopo la moltiplicazione si esegue una operazione di modulo l'overflow può essere evitato distribuendo la moltiplicazione su più parti e eliminando i termini più significativi (ovvero le cifre più a sinistra). Per fare ciò si può usare l'algoritmo di Schrange (vedi listato).
Ecco un estratto della classe, in linguaggo C++, che implementa il generatore presente su questo sito (si noti che la sequenza genera numeri reali tra 0 e 1 (1 escluso) in quanto divide il valore generato per $m$):

```
class CLCG {
public:
        long Seed, M, A;

        CLCG()
        {
                Seed = 1;
                M = 2147483647; // 2^31-1
                A = 1103515245;
        }

        double GetNextValue()
        {
                // ALGORITMO DI SCHRANGE
                long Q, Z, lo, hi, test;
                Q = M / A;
                Z = M % A;
                hi = Seed / Q;
                lo = Seed % Q;
                test = A*lo - Z*hi;
                Seed = (test > 0) ? test : test + M;
                return (double) Seed / M;
        }
};
```

e qui di seguito un breve programma che usa la classe precedente e con cui si possono riprodurre le sequenze di numeri generate da questo sito:

```
void main(void) {
        CLCG LCG;
        long min, max, n;
        cout << "Min?  "; cin >> min;
```

```
                        cout << "Max?  "; cin >> max;
                        cout << "Seme? "; cin >> LCG.Seed;

                        cout << "Quanti numeri? "; cin >> n;
                        cout << "ATTENZIONE: non si eliminano i duplicati"
<< endl;
                        while (n--)
                                cout << long(min + LCG.GetNextValue() *
(max-min) + 0.5) << endl;
                }
```