ESERCIZIO LEGGE DEI GRANDI NUMERI e NUMERI RANDOM CON I VETTORI

SOLUZIONE 1: UTILIZZO UN VETTORE PER TUTTI I NUMERI RANDOM

```
#include <iostream>
#include <ctime> //funzioni tempo per generare numeri random
#include <cstdlib> // contiene le funzioni per generare numeri random
using namespace std;
#define N 100000 // costante: fissare dimensione vettore numeri random
#define RANGE 6 // costante: fissare il range dei valori per gli elementi
int main()
{
  srand(time(0));   // seme di partenza prendendo il tempo della CPU
  int num1 = 0;
  int num2 = 0;
  int num3 = 0;
  int num4 = 0;
  int num5 = 0;
  int num6 = 0;

  int vet[N];   // dichiarazione vettore

  for(int i = 0; i<N;i++)
  {
    vet[i] = (rand() % RANGE) + 1;   // popolazione vettore
    if(vet[i] == 1) num1++;
    if(vet[i] == 2) num2++;
    if(vet[i] == 3) num3++;
    if(vet[i] == 4) num4++;
    if(vet[i] == 5) num5++;
    if(vet[i] == 6) num6++;
  }

  cout << "Ecco le statistiche finali: " << endl;
  cout << "Numero 1 uscito: " << num1 << endl;
  cout << "Numero 2 uscito: " << num2 << endl;
  cout << "Numero 3 uscito: " << num3 << endl;
  cout << "Numero 4 uscito: " << num4 << endl;
  cout << "Numero 5 uscito: " << num5 << endl;
  cout << "Numero 6 uscito: " << num6 << endl;
  return 0;
}
```

SOLUZIONE 2: UTILIZZO UN VETTORE PER TUTTI I NUMERI RANDOM e UN VETTORE PER MEMORIZZARE LE FREQUENZE

N.B. = non mi servono più 6 variabili per memorizzare i 6 numeri (salvo in un vettore)

```
#include <iostream>
#include <ctime> //funzioni tempo per generare numeri random
#include <cstdlib> // contiene le funzioni per generare numeri random
using namespace std;
#define N 100000 // costante: fissare dimensione vettore numeri random
#define RANGE 6 // costante: fissare il range dei valori per gli elementi
int main()
{
  srand(time(0));   // seme di partenza prendendo il tempo della CPU

  int vet[N];   // dichiarazione vettore numeri random
  int vetFrequenze[6];  // dichiarazione vettore frequenze
  for(int i = 0; i<6; i++)
  {
      vetFrequenze[i]=0;  // inizializzo a 0 le frequenze di uscita dei 6 numeri
  }
  for(int i = 0; i<N;i++)
  {
    vet[i] = (rand() % RANGE) + 1;   // popolazione vettore
    for(int j = 0; j<6; j++)
    {
      if(vet[i] == j+1)
      {
          vetFrequenze[j]=vetFrequenze[j]+1; // aggiorno le frequenze
      }
    }
  }

  cout << "Ecco le statistiche finali: " << endl;
  // stampo il vettore di frequenze
  for(int j = 0; j<6; j++)
  {
      cout << "Numero " << j+1 << " uscito: " << vetFrequenze[j] << endl;
  }

  return 0;
}
```

SOLUZIONE 3: UTILIZZO UN VETTORE PER TUTTI I NUMERI RANDOM e UN VETTORE PER MEMORIZZARE LE FREQUENZE – VERSIONE COMPATTATA

```
#include <iostream>
#include <ctime> //funzioni tempo per generare numeri random
#include <cstdlib> // contiene le funzioni per generare numeri random
using namespace std;
#define N 100000 // costante: fissare dimensione vettore numeri random
#define RANGE 6 // costante: fissare il range dei valori per gli elementi
int main()
{
  srand(time(0));   // seme di partenza prendendo il tempo della CPU

  int vet[N];   // dichiarazione vettore numeri random
  // dichiarazione vettore frequenze inizializzate a 0 per le frequenze di uscita dei 6 numeri
  int vetFrequenze[6]={0};  // versione compattata di inizializzazione a 0

  for(int i = 0; i<N;i++)
  {
    vet[i] = (rand() % RANGE);   // popolazione vettore
    vetFrequenze[vet[i]]++; // aggiorno le frequenze
  }
  cout << "Ecco le statistiche finali: " << endl;
  // stampo il vettore di frequenze
  for(int j = 0; j<6; j++)
  {
      cout << "Numero " << j+1 << " uscito: " << vetFrequenze[j] << endl;
  }

  return 0;
}
```