```
#include "TH1.h"
#include "TF1.h"
void rChi2()
{
//https://root.cern.ch/doc/master/classTRandom1.html
//https://root.cern.ch/doc/master/TRandom1_8cxx_source.html
//The algorithm for this random engine has been taken from the
//original implementation in FORTRAN by Fred James as part of
//CLHEP.
//The initialisation is carried out using a Multiplicative
//Congruential generator using formula constants of L'Ecuyer as
// described in "F.James, Comp. Phys. Comm. 60 (1990) 329-344".
//https://root.cern.ch/doc/master/classTRandom2.html
//https://root.cern.ch/doc/master/TRandom2_8cxx_source.html
//Random number generator class based on the maximally
//quidistributed combined Tausworthe generator by L'Ecuyer.
//The period of the generator is 2**88 (about 10**26) and it
//uses only 3 words for the state.
//http://root.cern.ch/root/html/TRandom3.html
//https://root.cern.ch/doc/master/TRandom3_8cxx_source.html
//TRandom3, is based on the "Mersenne Twister generator", and is the recommended
one, since it has good random proprieties (period of about 10**6000 ) and it is
fast
        // create random number generator

  gRandom = new TRandom2(0);
  int Ntry;
  cout << "Ntry=";
  cin >> Ntry;

  int NN;
  NN=1000;

  TH1F *h = new TH1F("h","bin3",40,0,40); // NN/10
//-------Ntry------------------

  for(int j=1; j<Ntry; j++) {

     double mean=0.0;
     double variance=0.0;
     double chi2=0.0;
     double f[10];
     for(int i=0; i<10; i++) {
       f[i]=0.;
     }


     for(int i=1; i<NN; i++) {
       double fx=gRandom->Uniform(0,1);
       int k=int(10*fx)%10;
       f[k]+=1.0;
       mean+=fx;
       variance+=fx*fx;
     }

     mean=mean/double(NN);
     variance=sqrt(variance/double(NN)-mean*mean);

     for(int i=0; i<10; i++) {
//        cout<<i<<" "<<f[i]/double(NN)<<endl;
        chi2+=(f[i]-double(NN)/10.0)*(f[i]-double(NN)/10.0)/(double(NN)/10.0);
     }
     chi2=chi2;
//     cout<<"mean= "<<mean<<endl;
//     cout<<"variance= "<<variance<<endl;
//     cout<<"chi2= "<<chi2<<endl;
```

```
     h->Fill(chi2,1.0/double(Ntry));
  }
//-------Ntry------------------

     h->Draw();

     TF1 *f1 = new TF1("chi-square distribution",ChiSquareDistr,0.,40,1);
     f1->SetParameter(0,9.0);
     f1->SetLineColor(49);
     f1->Draw("AC same");


}


Double_t ChiSquareDistr(Double_t *x,Double_t *par)
{
// Chisquare density distribution for nrFree degrees of freedom
Double_t nrFree = par[0];
Double_t chi2 = x[0];
if (chi2 > 0) {
  Double_t lambda = nrFree/2.;
  Double_t norm = TMath::Gamma(lambda)*TMath::Power(2.,lambda);
  return TMath::Power(chi2,lambda-1)*TMath::Exp(-0.5*chi2)/norm;
} else
return 0.0;
}
```