////////////////////////////RESULTADOS DEL EJERCICIO 2.6//////////////////////////////////////

********** RESULTADOS PARA AMBOS METODOS CON Nmuestreo = 500 millones **********

Tiempo_Muestreo-Uniforme = 49.73 segundos
resultado de la integral[muestreo uniforme] = 0.636619 || varianza[muestreo uniforme] = 0.0947094

Tiempo_Muestreo-de-Importancia = 0.01 segundos
resultado de la integral[muestreo de importancia] = 0.636723 || varianza[muestreo de importancia] = 0.000982833

Comparacion varianzas[uniforme/de importancia] = 96.3636
Eficiencia[uniforme/de importancia] = 479216

****

////////////////////////////CODIGO EN LENGUAJE C++//

```
#include <iostream>
#include <vector>
#include <math.h>
#include <iomanip>
#include <algorithm>
#include <stdio.h>
#include <gsl/gsl_rng.h>
#include <time.h>
#include <ctime>
using namespace std;



int main()
{

/* Se utiliza el metodo ranlux0 [se lo indica en la linea de comandos por medio de la variable GSL_RNG_TYPE].
   Descripcion del algoritmo ranlux0:
   PERIODO=10^171
   RAPIDEZ=571 k doubles/second
*/

// Definicion de las variables para generar numeros aleatorios.
  const gsl_rng_type * T;
  gsl_rng * r;
  gsl_rng_env_setup();
  T = gsl_rng_default;
  r = gsl_rng_alloc (T);

// Definicion de las variables genericas para ambos metodos
  long i,a,b,c,i1,Nq, Nq2;
  double u1,u2,g0,g1,r1,s1,r2,s2,resultado1,resultado2,T1,T2,x;

// Constantes.
  const double PI = 3.141592;
  a=0;
  b=1;
  c=1;
  Nq=500000000;

//*************************************************************************************************************************************
// Metodo de MUESTRO UNIFORME
  clock_t t1=clock();
  r1=0.;
  s1=0.;
  for (i=0;i<=Nq;i++)
    {
      u1=gsl_rng_uniform (r);
      g0=cos(PI*(a+(b-a)*u1)/2.);
      r1=r1+g0;
      s1=s1+g0*g0;
    }
  r1=r1/Nq;
  s1=pow((s1/Nq-r1*r1),0.5);
  resultado1=(b-a)*r1;
  clock_t t2=clock();
  T1=double(t2-t1)/CLOCKS_PER_SEC;
  cout<<endl<<"********** RESULTADOS PARA AMBOS METODOS CON Nmuestreo = 500 millones **********";
  cout<<endl;
  cout<<endl<<"Tiempo_Muestreo-Uniforme = " << T1 << " segundos";
  cout<<endl<<"resultado de la integral[muestreo uniforme] =  "<<resultado1<<" || "<<"varianza[muestreo uniforme] = "<<pow(s1,2);
  cout<<endl;
//*************************************************************************************************************************************
// Metodo de MUESTRO DE IMPORTANCIA
  Nq2=ceil(pow(0.00099062/0.094719,2)*Nq);
  clock_t t3=clock();
  r2=0.;
  s2=0.;
  for (i=0;i<=Nq2;i++)
```

```
    {
      u2=gsl_rng_uniform (r);
      x=2*cos((acos(u2)+PI)/3.);
      g1=cos(PI*x/2.)/(1.5*(1-x*x));
      r2=r2+g1;
      s2=s2+g1*g1;
    }
  resultado2=r2/Nq2;
  s2=pow((s2/Nq2-resultado2*resultado2),0.5);
  clock_t t4=clock();
  T2=double(t4-t3)/CLOCKS_PER_SEC;
  cout<<endl<<"Tiempo_Muestreo-de-Importancia = " << T2 << " segundos";
  cout<<endl<<"resultado de la integral[muestreo de importancia] = "<<resultado2<<" || "<<"varianza[muestreo de importancia] =
"<<pow(s2,2);
  cout<<endl;
  cout<<endl<<"Comparacion varianzas[uniforme/de importancia] = "<<pow(s1,2)/pow(s2,2);
  cout<<endl<<"Eficiencia[uniforme/de importancia] = "<<T1*pow(s1,2)/(T2*pow(s2,2))<<endl;
  cout<<endl<<"**********************************************************************************************";
  cout<<endl;

}
```