GENERACION DE NUMEROS DE DIFERENTES BITS

Alumno: Carlos Gabriel Morales Umasi

Curso: Algebra Abstracta

LIBRERIAS-CODIGO-GITHUB

```
#include <iostream>
#include <string>
#include <sstream>
#include <vector>
#include <random>
#include <bitset>
#include <windows.h>
#include <stdio.h>
#include <fstream>
#include <NTL/ZZ.h>
```

```
using namespace NTL;
using namespace std;


class RC4
{
private:ZZ A;
        ZZ B;
        ZZ semilla;
        ZZ semillabits;
public:
   string alfabeto = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
   ZZ bits = conv<ZZ>(1024);
   vector<string> K;vector<string> S;
```

```
   DWORD nicks=0;
   RC4(ZZ,ZZ);
   vector<string> secuencia_cifrante(vector<string>&);
   ZZ bits_();
   ZZ generarnumale();
   ZZ principal();ZZ stringtozz(string);ZZ mod(ZZ,ZZ);
   void iteraciones(vector<string>&);
   string zztostring(ZZ);string str_bin(string);
   int Z_z(ZZ);
};



RC4::RC4(ZZ _A,ZZ _B)
{
 semilla=generarnumale();
 semillabits=bits_ ();
 A=_A;
 B=_B;
}
ZZ RC4::stringtozz(string str)
{
 ZZ z(NTL::INIT_VAL, str.c_str());
return z;
}
string RC4::zztostring(ZZ num)
{
 stringstream buffer;
 buffer << num;
return buffer.str();
```

```
}
ZZ RC4::bits_()
{
   ZZ x,i,min,max,val1,val2;
   x=1;
   for(i=0;i<bits;i++)
   {x=x*2;}
   min=x/2;
   max=x-1;
   ZZ sem_;
   sem_=semilla;
   bool p=true,p1=true;
   string nani=zztostring(min);
   string te=zztostring(sem_);
   val1=nani.length();
   val2=te.length();
   if(val2<val1)
   {
    ZZ n,m;
    m=val1-val2;
    for(int jo=0,n=0;n<m;n++,jo++)
      {te=te+te[jo];}
    sem_=stringtozz(te);
   }
   else
   {while(p==true)
    {
    string tempe=zztostring(sem_);
    if(min<sem_ && sem_<max)
```

```
    {sem_=stringtozz(tempe);p=false; }
    else
    {tempe.erase(0,1);sem_=stringtozz(tempe);}
       p=false;
       i++;
    }
   }
   cout<<"Intervalo de "<<x/2<<"-"<<x-1 <<endl;
   return sem_;
}
ZZ RC4::generarnumale()
{
 HWND ventana;DWORD pid;HANDLE hp;
 char buffer[222];
 ventana=FindWindow(0,"Whatsapp");//necesita estar en 2do plano para sacar la informacion
 GetWindowThreadProcessId(ventana,&pid);
 hp=OpenProcess(PROCESS_ALL_ACCESS,true,pid);
 ReadProcessMemory(hp,(PBYTE*)nicks,&buffer,sizeof(buffer),0);
 ZZ x = conv<ZZ>(pid);//int a zz
return x;
}
int RC4::Z_z(ZZ num)
{
        string temp = zztostring(num);
        int numero = stoi(temp);
        return numero;
}
ZZ RC4::mod(ZZ a, ZZ b)
{
```

```
            if (a >= 0)
            {return a - (a / b) * b;}
            else
            {return a - ((a / b) - 1) * b;}
}

void RC4::iteraciones(vector<string> &K)
{
  ZZ f,i,Si,Ki,cuatro;
  string K_;
  string k_i;
  cuatro=A;//modulo a ();
  int temp,contador;
  K_=zztostring(semillabits);
  cout<<"K= "<<K_[7]<<endl;
  string pt;
  int tam_k=K_.length();
  for(i=0;i<A;i++)
  {
    if(contador>=tam_k)
    {contador=0;}
    temp=Z_z(i);
    string i_K=K[temp];
    pt=K_[contador];
    Si=stringtozz(i_K);
    Ki=stringtozz(pt);
       f=mod((f+Si+Ki),cuatro);
       contador=contador+1;
       //variables cambio
```

```
      k_i=K[temp];
      K[temp]=K[Z_z(f)];
      K[Z_z(f)]=k_i;
    }
}
string RC4::str_bin(string msj)
{string to_m, to_m_mem;
  int indi = alfabeto.find(msj[0]);
  for (int i = 0; i < msj.size(); i++)
  {
    indi = alfabeto.find(msj[i]);
    bitset<8> m(indi);
    to_m = m.to_string();
    to_m_mem += to_m;
  }
  return to_m_mem;
}
vector<string> RC4::secuencia_cifrante(vector<string> &S)
{
  ZZ I,modu,i,t,str_IF,F;
  string k_i;
  vector<string>::iterator indi=K.begin();
  for(i=0;i<A;i++)
    {
      I=mod(I+1,A);
      modu=mod(F+stringtozz(S[Z_z(i+1)]),A);
      k_i=S[Z_z(I)];
      S[Z_z(I)]=S[Z_z(F)];
      S[Z_z(F)]=k_i;
```

```
    str_IF=stringtozz(S[Z_z(I)])+stringtozz(S[Z_z(F)]);
    t=mod(str_IF,A);
    string oreimo=str_bin(S[Z_z(t)]);
    K.push_back(oreimo);
   }
 return K;
}
ZZ RC4::principal()
{
 cout<<semilla<<" "<<B<<" "<<A<<endl;
 cout<<"SEMILLA EN BITS: "<<semillabits<<endl;
 cout<<"S:"<<A<<endl;
 ZZ c,x;
 c=1;
 for(x=1;x<A+c;x++)
 {K.push_back(zztostring(x));}
 vector<string>::iterator indi=K.begin();
 for (indi; indi != K.end(); indi++) //vector l
  {cout << *indi <<","; }
 cout<<endl;
  //cout<<"marca de agua"<<endl;
 iteraciones(K);
 string str_ini;
 vector<string>::iterator iterador=K.begin();
 for (iterador; iterador != K.end(); iterador++) //vector l
 {str_ini+=(*iterador);
 cout << *iterador <<",";
 }
  cout<<endl;
```

```
    cout<<"SECUENCIA CIFRANTE: "<<endl;
    cout<<endl;
    vector<string>sec_cif;//final
   sec_cif=secuencia_cifrante(K);
    vector<string>::iterator itera2;
    for(itera2=sec_cif.begin(); itera2!=sec_cif.end(); itera2++)
    { cout<<*itera2; }
    cout<<endl;
    cout<<"NUMEROS DE N BITS: "<<endl;
    cout<<str_ini<<endl;
}
int main()
{
    ZZ a,b;
    a=250;b=5;
    RC4 p1=RC4(a,b);
    p1.principal();
return 0;
}
```

---

https://github.com/CarlosGabrielMoralesUmasi/ALGEBRA-ABSTRACTA-PARTE2