RSA CON BLOQUES

ALUMNO: Carlos Gabriel Morales Umasi

CURSO: ALGEBRA ABSTRACTA

Clase-funciones-main-github

```
#include <iostream>
#include <string>
#include <sstream>// para zz to string
#include <NTL/ZZ.h>
#include <vector>
#include <random>
#include <NTL/vector.h>
using namespace std;
using namespace NTL;
```

```
class RSA
{
        ZZ d, p, q, fi_N;
public:
        RSA(ZZ), RSA(ZZ, ZZ);
        ZZ mod2(ZZ, ZZ), mcd(ZZ, ZZ), inversa(ZZ, ZZ), potenciar(ZZ, ZZ), potenciaymod(ZZ, ZZ, ZZ),
randomprimo2(int);
        ZZ str_zz(string);
        ZZ e, n, max;
        int z_int(ZZ), mod1(int, int);
        string int_str(int), z_str(ZZ);
        string alfabeto = "ABCDEFGHIJKLMNOPQRSTUVWXYZ,.-(
)abcdefghijklmnopqrstuvwxyz<>*1234567890";
        string cif(string), descif(string),  add(string, ZZ), completar(string, int);};
```

```
---------------------------------------------------------------------------------------------------------------------------------

//contructor

RSA::RSA(ZZ max_bits)

{

        max = max_bits;

        //p = RandomPrime_ZZ(8);q = RandomPrime_ZZ(8);e = RandomPrime_ZZ(8);//no siempre genera los números en n bits

        //ZZ a;

        //GenPrime(p,15);

   p=17;//predeterminando p y q

        q=19;

        n = p * q;

        fi_N = (p - 1) * (q - 1);

        e=43;

        d=mod2(e, fi_N);

        d=inversa(d, fi_N);d=mod2(d, fi_N);

        cout << "P: " << p <<endl<< "Q: " << q <<endl<< "N: " << n <<endl;

        cout << "Fi de N: " << fi_N <<endl;

        cout << "Su clave publica es: " <<e<<endl;

        cout << "Su clave privada es: " <<d<< endl;

}

---------------------------------------------------------------------------------------------------------------------------------

RSA::RSA(ZZ nrecept, ZZ clave_publica)

{

        e = clave_publica;

        n = nrecept;

}
```

```
int RSA::mod1(int a, int b)
{
	if (a >= 0){return a - (a / b) * b;}
	else{return a - ((a / b) - 1) * b;}
}
ZZ RSA::mod2(ZZ a, ZZ b)
{
	if (a >= 0){return a - (a / b) * b;}
	else{return a - ((a / b) - 1) * b;}
}
ZZ RSA::mcd(ZZ a, ZZ b)
{
	ZZ r;
	while (true)
  {r = mod2(a, b);
		if (r == 0){return b;}
		if (r > (b / 2)){     r = b - r;}
		a = b;
		b = r;
	}
}
//inversa
ZZ RSA::inversa(ZZ nom, ZZ modulonum)
{
  ZZ modulo_cero=modulonum,variable,cociente;
	ZZ variable0, variable1;
	variable0 = 0, variable1 = 1;
	while (nom > 1) {
		cociente=nom/modulonum;
```

```
                variable=modulonum;
                modulonum=mod2(nom, modulonum);
                nom=variable;
                variable=variable0;
                variable0=variable1 - cociente*variable0;
                variable1=variable;
        }
        if (variable1 < 0)          {variable1 += modulo_cero;      }
        return variable1;
}
//potencia
ZZ RSA::potenciar(ZZ num_base, ZZ elevar)
{
        ZZ total, i;
        total = 1; i = 0;
        for (i; i < elevar; i++){total *= num_base;}
        return total;
}
ZZ RSA::potenciaymod(ZZ num_base,ZZ elevado, ZZ modulo)
{ZZ rta_exp,enter;
   rta_exp=1;enter=2;
   while(elevado!=0)
   {if(mod2(elevado,enter)==1){rta_exp=mod2(rta_exp*num_base,modulo);}
     num_base=mod2(num_base*num_base,modulo);
     elevado=elevado/enter;
   }
   return rta_exp;
}
```

```
int RSA::z_int(ZZ num)//de zz a entero
{
	string temp = z_str(num);
	int numero = stoi(temp);
	return numero;
}
string RSA::int_str(int a)//de entero a string
{
	ostringstream temp;
	temp << a;
	return temp.str();
}
string RSA::z_str(ZZ num)//de zz a string
{
	stringstream convertido;
	convertido<<num;
	return convertido.str();
}
ZZ RSA::str_zz(string str)//de string a zz
{
	ZZ zz_(NTL::INIT_VAL, str.c_str());
	return zz_;
}
**************************************************************************
//añadir ceros
string RSA::add(string tamm, ZZ max)
{
	string str_fin;
	ZZ espacio = ZZ(tamm.size());
```

```
        espacio=max-espacio;
        for (int i = 0; i < espacio; i++){str_fin += "0";}
        str_fin += tamm;
return str_fin;
}
//CIFRADO
string RSA::cif(string msj)
{
        string txt, conv;
 string temp, temp2;
 string igua, igua2;
        ZZ pos, tam;
 ZZ pos2;
        tam = alfabeto.size();
        for (int i = 0; i < msj.size(); i++)
 {pos = alfabeto.find(msj[i]);
                igua = z_str(pos);
                igua2 = z_str(tam);
                conv += add(igua, ZZ(igua2.size()));
        }string ntemp = z_str(n);
 int bloq = ntemp.size() - 1;
        conv = completar(conv, bloq);
        for (int i = 0; i < conv.size(); i += bloq)
 {for (int j = 0; j < bloq; j++){temp += conv[i+j]; }
                pos2 = str_zz(temp);
                pos2 = potenciaymod(pos2, e, n);
                temp2 = z_str(pos2);
                temp2 = add(temp2,ZZ(ntemp.size()));
                txt += temp2;
```

```
                temp.clear();
                temp2.clear();
        }
        return txt;
}
//VERIFICACION DE CEROS
string RSA::completar(string msj, int division)
{
string dev;
int size = msj.size();
        size = mod1(size, division);
        size = division - size;
if (mod1(size, 2) == 0)
 {
                for (int i = 0; i < size / 2; i++){msj += "25"; }
        }
if (mod1(size, 2) != 0)
 {
                for (int i = 0; i < size / 2; i++){      msj += "25"; }
                msj += "7";
        }
        return msj;
}
//descifrado no terminado
/*string RSA::descif(string cif)
{
        string txt, conv, temp, temp2;
        ZZ pos, tam, pos2;
        string ntemp = z_str(n);
```

```
        int i_;int bloq = ntemp.size();
        string descif;
        for (int i = 0; i < cif.size(); i += bloq)
 {
                for (int j=0;j<bloq;j++){temp += cif[i+j];   }
                pos2=str_zz(temp);
                pos2=potenciaymod(pos2, d, n);
                temp2=z_str(pos2);tam=ZZ(bloq-1);
                temp2=add(temp2,tam);
                conv+=temp2;
                temp.clear();
                temp2.clear();
        }
        int tamabc = alfabeto.size();
        string tama = int_str(tamabc);
        tamabc = tama.size();
        for (int i = 0; i < conv.size(); i += tamabc)
 {
                for (int j = 0; j < tamabc; j++) {temp += conv[i + j];           }
                i_=stoi(temp);
                temp2=alfabeto[i_];
                descif+=temp2;
                temp.clear();
        }
        return descif;
}
*/
```

```
**************************************************************************
int main()
{
	ZZ max(5);
	RSA emisor(max);
	ZZ n, clave;clave =43 , n =1003;
	RSA recept(n, clave);
	string msj = "hola mundo";
	string cif = recept.cif(msj);//string descif = emisor.descif(cif);
	cout << "CIFRADO : " << cif ;//<< endl<< "DESCIFRADO : " << descif << endl;
}
```

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