Nombre: Cuitpa Mamani, Joaquin Esteban

Github: https://github.com/xnabux/Algebra-abstracta.git

Nota : Falta mejorar

```
#include <iostream>
#include <string>
#include <stdlib.h>
#include <time.h>
#include <vector>
#include <math.h>

using namespace std;

using namespace std;
class rsa
{
public:
	//rsa
	rsa();
	rsa(int, int);
	string cifrar(string);
	string descifrar(string);

	//mat
	int mod(int, int);
	int valor_absoluto(int);
	int euclide_exten(int, int);
	int euclide(int, int);
	int inversa_mul(int, int);

	//expo
	int exponencial(int, int, int);

	//otros
	int aleatorio(int, int);

	//variables
	int n;
	int e;
	string abc = "ABCDEFGHIJKLMNOPQRSTUVWXYZ,.-(
)abcdefghijklmnopqrstuvwxyz<>*1234567890";


	int d;
	int fi_de_N;
private:

};
```

```
#include "rsa.h"

rsa::rsa(/*int n, int e*/) {//que seria n,e????
	int p = 17;//aqui tengo que usar la criba
```

```
	int q = 43;//aqui tengo que usar la criba

	n = p * q;
	fi_de_N = (p - 1) * (q - 1);

	do {
		e = aleatorio(fi_de_N, 1);//falta agragar funcion aleatorio
	} while (euclide(e, fi_de_N) != 1); //si es = 1 entonces sale

	d = inversa_mul(e, fi_de_N);

	cout << "\Clave publica Ce = <N,e>: <" << n << "," << e << ">";
	cout << "\Clave privada Cd = <N,d>: <" << n << "," << d << ">";
}

string rsa::cifrar(string mensaje) {
	string mensaje_cifrado;
	for (int i = 0; i < mensaje.size(); i++) {
		int pos = abc.find(mensaje.at(i));
		int x = exponencial(pos, e, n); //ver como funciona la exponencial

		string valor = to_string(x);//que sto_string
		mensaje_cifrado += valor;
		mensaje_cifrado += " ";

	}
	return mensaje_cifrado;
}

string rsa::descifrar(string mensaje) {//revisar bien como es!!!!!!!
	string mensaje_descifrado;

	for (int i = 0; i < mensaje.size(); i++) {
		int pos_esp = mensaje_descifrado.find(" ", i);
		string letra = mensaje_descifrado.substr(i, pos_esp);//que es substr
		mensaje.erase(i, letra.size() + 1);
		int q = stoi(letra);//que es stoi
		int D = exponencial(q,rsa::d,n);
		mensaje_descifrado += abc[D];
	}
	return mensaje_descifrado;
}


//funciones matematicas
int rsa::aleatorio(int rango, int inicio) {
	int num = inicio + rand() % (rango - inicio);
	return num;
}

int rsa::mod(int a, int b) {
	int r = a - ((a / b) * b);
	if (r < 0)
		r = a - (((a / b) - 1) * b);
	return r;
}

int rsa::valor_absoluto(int a) {
```

```
	if (a < 0) {
		a = a * -1;
	}
	return a;
}

int rsa::euclide(int a, int b) {
	if (valor_absoluto(b) > valor_absoluto(a))
		return euclide(b, a);
	if (b == 0) {
		return a;
	}
	if (mod(a, 2) == 0 && mod(b, 2) == 0) {
		return 2 * euclide(a / 2, b / 2);
	}
	if (mod(a, 2) == 0 && mod(b, 2) == 1) {
		return euclide(a / 2, b);
	}
	if (mod(a, 2) == 1 && mod(b, 2) == 0) {
		return euclide(a, b / 2);
	}
	return euclide((valor_absoluto(a) - valor_absoluto(b)) / 2, b);
}

int rsa::euclide_exten(int a, int n) {//revisar el euclides extendido
	int s1 = 1;
	int s2 = 0;
	while (n > 0) {
		int q = a / n;
		int r = a - q * n;
		a = n;
		n = r;
		int s = s1 - q * s2;
		s1 = s2;
		s2 = s;
	}
	return s1;
}

int rsa::inversa_mul(int a, int n) {//ver como funciona la inversa multiplicativa
	if (euclide(a, n) == 1) {
		int x = euclide_exten(a, n);
		if (x < 0) {
			x = x + n;
		}
		return x;
	}
	else {
		cout << "No tiene inversa\n";
	}
}

int rsa::exponencial(int b, int e, int n) {//ver como funciona la exponencial
	int pow = 1;
	do {
		if (mod(e, 2) != 0) {
			pow = mod(b * pow, n);
		}
```

```
            b = mod(b * b, n);
            e /= 2;
        } while (e != 0);
        return pow;
}
```

```
#include "rsa.h"



int main() {
        rsa llamar;

        string mensaje;
        cout << "\nEscribe tu mensaje: ";
        getline(cin,mensaje);

        string mensaje_cifrado = llamar.cifrar(mensaje);
        cout << "\nTu mensaje cifrado es: " << mensaje_cifrado;

        string mensaje_descifrado = llamar.descifrar(mensaje_cifrado);
        cout << "\nTu mensaje descifrado es: " << mensaje_descifrado;



        return 0;
}
```