RSA FINAL

Curso: ALGEBRA ABSTRACTA

Alumno: CARLOS GABRIEL MORALES UMASI

Orden de presentación: Class-Functions-Main-Github(link)

```
class RSA
{
   ZZ p,q,d,e_pub,n_pub, e,n,phi,tam; //valores dados  RSA
   public:
      RSA(int ,ZZ ,ZZ );
      RSA(int);
      RSA(ZZ,ZZ,ZZ,ZZ,ZZ,ZZ);
      vector<ZZ>  euclides_extendido(ZZ,ZZ);
      string alfabeto= "ABCDEFGHIJKLMNOPQRSTUVWXYZ,#-(
)abcdefghijklmnopqrstuvwxyz<>*1234567890";
      string cifrar(string); //función cifrar
      string descifrar(string);//función descifrar
      ZZ modulo(ZZ,ZZ);//función modulo
      ZZ euclides(ZZ,ZZ);//función euclides
      ZZ inversa_RSA(ZZ ,ZZ);//función inversa
      ZZ multiplicacion_modular(ZZ , ZZ ,ZZ );//multiplicación modular para ZZ
      ZZ exponenciacion_modular(ZZ ,ZZ ,ZZ );//exponenciación modular para ZZ
      ZZ phi_euler(ZZ p,ZZ q);//función PHI DE EULER
      ZZ str_zz(string );//convertir de string a ZZ
      string rellenado_0(int);//función llenar con ceros
      ZZ chau_ceros(string);//función eliminar ceros
      string rellenado_aux(string,int);//función para los espacios extras (en este caso utilize el '#' )
      string rellenado_2(string,string);//función para seguir luego de completer los ceros
      string rellenado_descifrado(string , int);//función rellenado para poder poder descifrar
```

```
        string zz_str(ZZ);//función de ZZ a string
        string int_str(int a);//función de int a ZZ
        int zz_int(ZZ );//función ZZ a int
        int str_int(string );//función string a int
        ZZ aleatorio_prim(int );//función para que sean primos los aletorios en bits
        void generar_claves(int);//función generar claves con los aleatorios
};
```

Funciones:

```
zz_str(ZZ z) {
   stringstream a;
   a << z;
   return a.str();
}
str_zz(string str){
   ZZ numer(INIT_VAL, str.c_str());
   return numer;
}
int_str(int a)
{
   stringstream bleach;
   bleach << a;
   string aux=bleach.str();
   return aux;
}
str_int(string a)
{
   stringstream bleach(a);
   int aux = 0;
```

```
   bleach >> aux;
   return aux;
}
zz_int(ZZ numer)
{
   int n;
   conv(n,numer);
   return n;
}
aleatorio_prim(int bits)
{
    ZZ a;
    GenPrime(a,bits);
    return a;
}
RSA(int bits)
{
   generar_claves(bits);
   e_pub=e;
   n_pub=n;
   cout<<"E_DEL_MEN: "<<e_pub<<endl;
   cout<<"N_DEL_MEN: "<<n_pub<<endl;
}
RSA(int bits, ZZ e2, ZZ n2)
{
   generar_claves(bits);
   e_pub=e2;
   n_pub=n2;
}
```

```
generar_claves(int bits)
{
   ZZ min;
   min = (2^bits)/2;
   p=aleatorio_prim(bits);
   cout<<"P:"<<p<<endl;
   q=aleatorio_prim(bits);
   cout<<"Q:"<<q<<endl;
   n=p*q;
   cout<<"N: "<<n<<endl;
   phi=phi_euler(p,q);
   cout<<"PHI: "<<phi<<endl;
   e=aleatorio_prim(bits);
   cout <<"E:"<<e<<endl;
   tam=alfabeto.length();
   d=inversa_RSA(e,phi);
   cout <<"D:"<<d<<endl;
}
modulo(ZZ a,ZZ b)
{
   ZZ q=a/b;
   ZZ r=a-q*b;
   if(r<0){r+=b;}
   return r;
}
euclides(ZZ a, ZZ b)
{
   ZZ res=modulo(a,b);
   while(res!=0)
```

```
    {
     a=b;
     b=res;
     res=modulo(a,b);
    }
    return b;
}
vector<ZZ> RSA::euclides_extendido(ZZ a, ZZ b)
{
    vector<ZZ> result;
    ZZ r1 = a, r2 = b;
    ZZ x1 = to_ZZ(1), x2 = to_ZZ(0);
    ZZ y1 = to_ZZ(0), y2 = to_ZZ(1);
    ZZ q , r , x , y;
    while(r2>0)
    {
        q = r1/r2;
        r = r1 - q*r2;
        r1 = r2;
        r2 = r;
        x = x1 - q*x2;
        x1 = x2;
        x2 = x;
    }
    result.push_back(x1);
    result.push_back(x2);
    result.push_back(euclides(a,b));
    return result;
}
```

```
inversa_RSA(ZZ a,ZZ b)
{
   vector<ZZ> temp;
   temp=euclides_extendido(a,b);
   if(temp[0]<0){return temp[0]+b;}
   else{return temp[0];}
}
multiplicacion_modular(ZZ a, ZZ b,ZZ mod)
{
   return modulo((modulo(a, mod) * modulo(b,mod)), mod);
}
exponenciacion_modular(ZZ a,ZZ b,ZZ mod)
{
   ZZ resultado = ZZ(1);
   while(b>0)
   {
      if((b&1)==1)//VERIFICA SI "B" ES IMPAR O NO
      {
         resultado = multiplicacion_modular(resultado,a,mod);
      }
      a=multiplicacion_modular(a,a,mod);
      b=b/2;
   }
   return resultado;
}
phi_euler(ZZ p,ZZ q)
{
   ZZ res;
   res = (p-ZZ(1))*(q-ZZ(1));
```

```
    return res;
}
rellenado_0(int a)
{
    string llenar = int_str(a);
    string treat_fill = int_str(alfabeto.length());
    while(llenar.length()<treat_fill.length())
    {
        llenar = '0' + llenar; //llenado inicial
    }
    return llenar;
}
chau_ceros(string w)
{
  ZZ piv;
  string aux1;
  int pos=w.find('0');
  while(pos!=-1)
  {
      w.erase(pos,1);
  }
  piv=str_zz(w);
  return piv;
}
rellenado_2(string piv,string w)
{
    while(piv.length()<w.length()){piv = '0' + piv; }
    return piv;
}
```

```
rellenado_aux(string a, int n_n)
{
   string var1="27";
   int i=0;
   char b;
   while(a.length()<n_n)
   {
      i=i%2;
      b=var1[i];
      a=a+b;
      i++;
   }
   return a;
}
rellenado_descifrado(string a, int n_n)
{
   while(a.length()<n_n)
   {
      a=  '0' + a;
   }
   return a;
}
```

Contructor:

```
RSA::RSA(ZZ p_1,ZZ q_1,ZZ e_1,ZZ d_1,ZZ e_2,ZZ n_2)
{
   p = p_1;
```

```
   q = q_1;
   e = e_1;
   d = d_1;
   n = p_1 * q_1;
   phi = (p-ZZ(1))*(q-ZZ(1));
   e_pub = e_2;
   n_pub = n_2;
   cout<<"P:"<<p<<endl<<"Q:"<<q<<endl<<"N: "<<n<<endl<<"PHI:
"<<phi<<endl<<"E:"<<e<<endl<<"D:"<<d<<endl;
   cout<<"E_DEL_MEN: "<<e_pub<<endl<<"N_DEL_MEN: "<<n_pub<<endl;
}
```

Cifrar:

```
string RSA::cifrar(string mensaje)
{
        string mensaje_cifrado;
        ZZ aux, p,pos_letra_cifrar;
        ZZ pos_letra_cifrada;
   string mensaje_rellenado_0,aux1,aux2,bloque;
   string aux3;
   int pos_letra,piv;
        for(int i=0;i<mensaje.length();i++)
   {
      pos_letra=alfabeto.find(mensaje[i]);
      aux1=rellenado_0(pos_letra);
      mensaje_rellenado_0=mensaje_rellenado_0+aux1;
        }
        cout<<"Mensaje rellenado\n";cout<<mensaje_rellenado_0<<endl;
   aux2=zz_str(n); //valor de n en string para poder hacer los bloques
```

```
  int cont=0;
  string bloque_vacio;
  for(int i=0;i<(mensaje_rellenado_0.length()/(aux2.length()-1)+1);i++)
  {
    for(int j=1;j<aux2.length();j++)
    {
      if(cont<mensaje_rellenado_0.length())
      {
        bloque=bloque+mensaje_rellenado_0[cont];
        cout<<endl<<"Bloques:"<<endl;
        cout<<endl<<bloque<<" "<<endl;
        cont++;
      }
    }
    if(bloque.length()!=(aux2.length()-1))
    {
      bloque=rellenado_aux(bloque,aux2.length()-1);
    }
    pos_letra_cifrar=str_zz(bloque);
    pos_letra_cifrada=exponenciacion_modular(pos_letra_cifrar,e,n); //CIFRANDO
    string p_string=zz_str(pos_letra_cifrada);
    aux3=rellenado_2(p_string,aux2);
    mensaje_cifrado=mensaje_cifrado+aux3;
    bloque=bloque_vacio;
  }
  return mensaje_cifrado;
}
```

Descifrar:

```
string RSA::descifrar(string w)
```

```
{
   string mensaje_descifrado_bloques,mensaje_descifrado;
   string aux2=zz_str(n);
   string bloque,bloque_vacio;
   string aux1;
   char letter;
   int pos,pos1;
   ZZ pos_letra_descifrar,pos_letra_descifrada,p_letra_des;
   int cont=0;
   for(int i=0;i<w.length()/(aux2.length());i++)
         {
         for(int j=0;j<aux2.length();j++)
            {
            if(cont<w.length()){
               bloque=bloque+w[cont];
               cont++;
            }
         }
         //cout<<endl<<"Bloque en descifrado: "<<bloque<<endl;
         pos_letra_descifrar=str_zz(bloque);
         pos_letra_descifrada=exponenciacion_modular(pos_letra_descifrar,d,n);
         string p_string;
         p_string=zz_str(pos_letra_descifrada);
         aux1=rellenado_descifrado(p_string,aux2.length()-1);
         mensaje_descifrado_bloques=mensaje_descifrado_bloques+aux1;
         cout<<endl<<"MENSAJE DESCIFRADO BLOQUES: "<<mensaje_descifrado_bloques<<endl;
         bloque=bloque_vacio;
   }
   cont=0;
```

```
   for(int k=0;k<mensaje_descifrado_bloques.length()/2;k++){
      for(int l=0;l<2;l++)
      {
         if(cont<w.length()){bloque=bloque+mensaje_descifrado_bloques[cont];}
         cont++;
      }
      p_letra_des=str_zz(bloque);
      pos=zz_int(p_letra_des);
      cout<<endl<<"pocisiones de las letras:"<<pos<<endl;
      mensaje_descifrado=mensaje_descifrado+alfabeto[pos];
      bloque=bloque_vacio;
   }
   return mensaje_descifrado;
}
```

MAIN:

```
int main()
{
   RSA emisor(1024);
   ZZ N=to_ZZ(1),e,pos;
   string msj_cif,mensaje_descifrado;
   msj_cif=emisor.cifrar("ABSTRACTA");
   cout<<endl<<endl<<"MENSAJE CIFRADO: "<<endl<<msj_cif<<endl;
   mensaje_descifrado=emisor.descifrar(msj_cif);
   cout<<endl<<endl<<"MENSAJE DESCIFRADO: "<<mensaje_descifrado;
}
```

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

Incluye todos los códigos hechos desde el parcial hasta el final

Adjunto el link de la primera parte que va desde inicio de semestre hasta el parcial:

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