LISTA DE CONTENIDO

1. LISTA DE FIGURAS…………………………………………………………………2
2. LISTA DE CÓDIGOS………………………………………………………………..2
3. LISTA DE TABLAS…………………………………………………………………..2
4. DESARROLLO……………………………………………………………………….2
   1. MULTIPLICACION MODULAR………………………,……………………….2
   2. INVERSA MODULAR…………………………………….…………………….3
   3. EXPONENCIACION MODULAR…………………….………………………..5
5. CONCLUSIONES…………………………………………………………………….7
6. REFERENCIAS ……………………………………………………………………..7
7. LISTA DE FIGURAS

Figura 1: Resultado de Euclides…………………………………………………….3

Figura 2: Resultado de Euclides Extendido………………………………………..4

Figura 2: Ecuaciones para coeficientes de Bezout……………………………….4

Figura 3: Resultado de Algoritmo de Diofantes……………………………………6

1. LISTA DE CÓDIGOS

Codigo 1: Algoritmo de Euclides. ……………………………………………………2

Codigo 2: Algoritmo de Euclides Extendido. ……………………………………….3

Codigo 3: Algoritmo de Diofantes. …………………………………………………..6

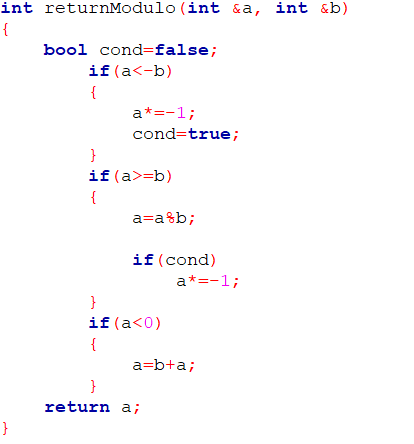
1. LISTA DE TABLAS

Tabla 1: Algoritmo de Euclides Extendido…………………………………………..5

1. DESARROLLO

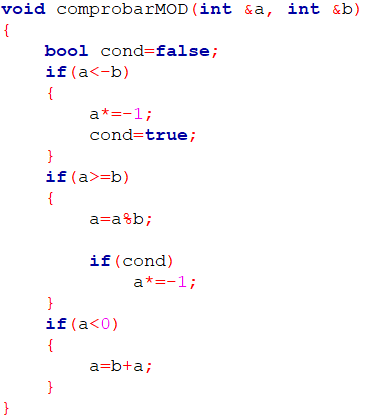
Para el desarrollo de esta práctica se utilizaron 3 funciones y una clase “Modulo”

Donde se sobre escribió el operador “\*” y se creó dos funciones: “Inversa” y “Exponente”.



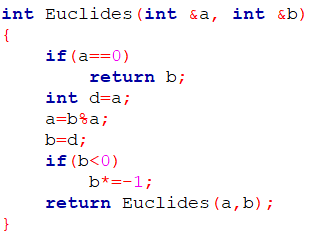
**Código 1: ReturnModulo.**

La función “ReturnModulo”, de tipo int con parámetros “a” y “b” que los pasa por referencia, comprueba si el numero supera al módulo o es menor que este, de cumplirse alguna de las restricciones hará las operaciones necesarias para que el numero sea menor que el módulo, comprobando si el número es negativo o mayor que el modulo retornando el número.



**Código 2: ComprobarMOD.**

La segunda función llamada “ComprobarMOD” hace lo mismo que la función “ReturnModulo” con la excepción de que la función es de tipo void y no devuelve nada.



**Código 3: Euclides.**

El código 3 implementa el algoritmo de Euclides ya desarrollado en clases pasadas.

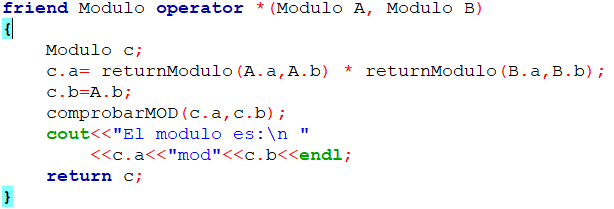
* 1. MULTIPLICACION MODULAR

Primero desarrollamos la multiplicación modular dentro de nuestra clase “Modulo” sobre escribiendo el operador “\*” pasando por parámetro dos objetos de tipo “Modulo”; creamos un nuevo objeto con el nombre “c” y realizamos la operación correspondiente:



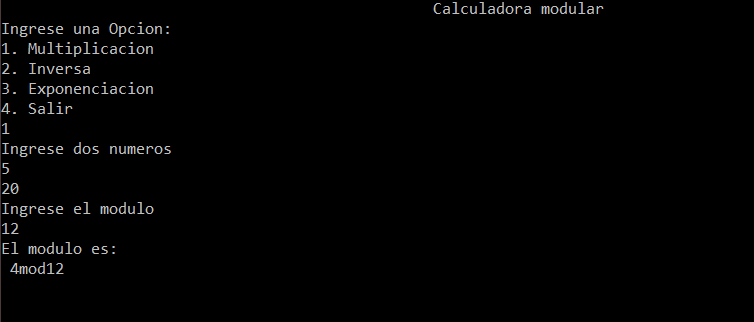
**Figura 1: Multiplicación Modular.**

Una vez aplicada la formula anterior, devolvemos el objeto “c” como se demuestra en el código siguiente:



**Código 4: Multiplicación modular.**

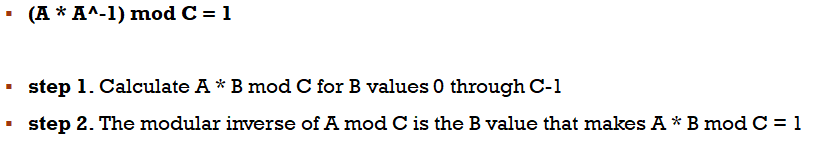
Al ejecutar el código, el programa nos pedirá ingresar una de las operaciones registradas en la interfaz, después dos números enteros y su modulo, pasando a mostrar la respuesta:



**Figura 2: Resultado de la multiplicación modular.**

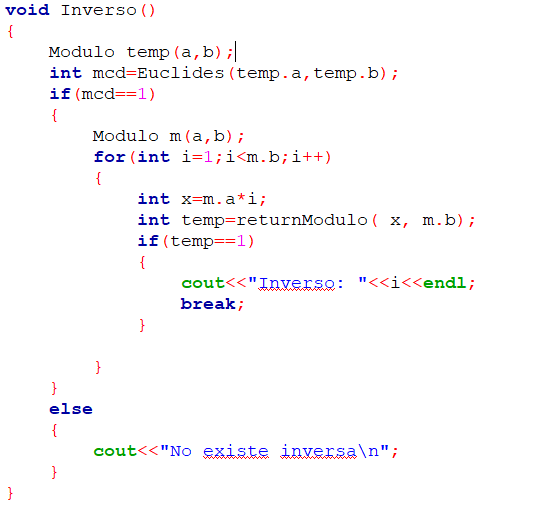
* 1. INVERSA MODULAR

Después pasamos a desarrollar la inversa modular que permite hallar, como dice el nombre, la inversa de un número con respecto al módulo en dos pasos:



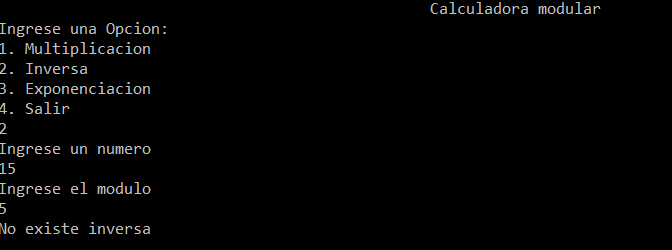
**Figura 3: Inversa Modular.**

Como dice la figura 3 primero calculamos multiplicamos (A \* B) mod C desde 0 hasta C-1. Después elegimos el B cuya multiplicación modular de “1”.



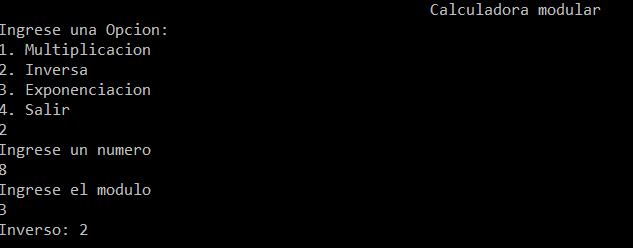
**Código 5: Inversa Modular.**

En el anterior código creamos un objeto tipo módulo de nombre “temp” y en una variable “mcd” almacenamos el ultimo residuo de por medio del algoritmo de Euclides, si este es igual a 1 entonces tiene inversa y procedemos a calcularla; hacemos un bucle “for” desde 1(Porque 0 jamás podrá ser la inversa de ningún numero) hasta m.b-1(Modulo), en una variable “a” multiplicamos m.a(el numero ingresado) por i(contador, en este caso viene a ser “B”) si el módulo de esa multiplicación viene a ser 1, entonces imprimimos el valor y hacemos un “break” ya que ya se ha encontrado la inversa y no es necesario más vueltas en el bucle.



**Figura 4: Resultado de un Numero sin Inversa.**

Si ingresamos un numero cuyo su “mcd” sea diferente de 1, es decir que no tenga inversa, el programa pasara a mostrar “No existe inversa”.

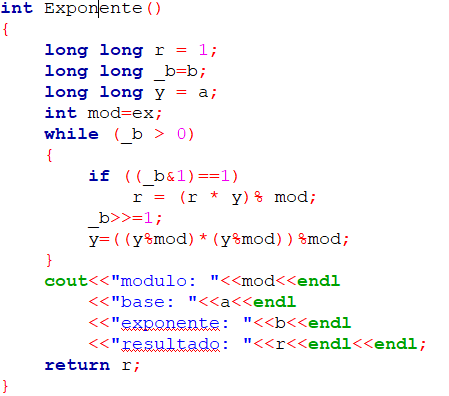


**Figura 5: Resultado de Inversa Modular.**

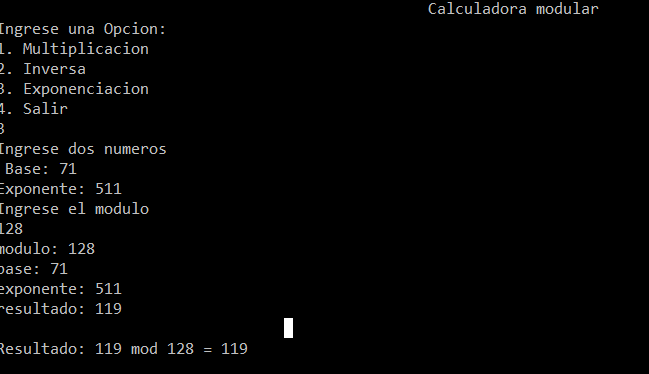
De caso contrario, si existe una inversa entonces pasara a ser mostrada en pantalla.

* 1. EXPONENCIACION MODULAR

Finalmente se realizo



**Código 6: Exponenciación Modular.**



**Figura 6: Resultado de Exponenciación Modular. []**

1. CONCLUSIONES

* Primero aprendimos el fruncimiento del algoritmo de Euclides que nos permite hallar el máximo común divisor (MCD) de una forma rápida y eficaz.
* Codificamos el algoritmo de Euclides Extendido con ayuda del algoritmo de Euclides y los coeficientes de Bezout.
* Finalmente descubrimos una forma de resolver una ecuación diofántica lineal de doble incógnita, a través de Euclides Extendido y MCD.
* Aplicamos la teoría de números y lo aprendido en clase por medio de la codificación de los algoritmos de Euclides, Euclides Extendido y Ecuaciones Diafonticas.
* Demostramos la validez de los algoritmos estudiados en clase en relación al cálculo del máximo común divisor comenzando por la codificación del algoritmo de Euclides usando recursividad.

1. REFERENCIAS

(s.f.).

Mordell, L. J. (1969). *Diophantine equations. Pure and Applied Mathematics .* Academic Press.

Shoup, V. (2008). *A Computational Introduction to Number Theory and Algebra.* Cambridge University Press.

von zur Gathen, J., & Gerhard, J. (2003). *Modern Computer Algebra.* Cambridge University Press.