Programación de computadores 2018-I

Ricardo Andrés Calvo Méndez
Documento: 1000861255

Mayo 2018

1. La Granja.

En una granja se crían un número de v - Vacas, a - Aves (pollos y gallinas) y e - escorpiones. Las vacas estan encerradas en un corral de nm metros cuadrados, las aves en un galpón y los escorpiones en vitrinas.

Problema 1. *La leche.*

Si una vaca necesita m metros cuadrados de pasto para producir x litros de leche, ¿Cuántos litros de leche se producen en la granja?

granja.h : Linea 4.

granja.cpp : Lineas 3-5.

Modelo Matemático: La relación de la leche con los metros cuadrados de pasto es directamente proporcional; para ello se establecen las siguientes variables:

$x :=$ Leche producida por m metros cuadrados de pasto.

$m :=$ Metros cuadrados de pasto que se requieren para producir x litros de leche.

$t :=$ Metros cuadrados totales de pasto.

Entonces:

$$Leche : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$
$$(m, x, t) \mapsto \left\{ \frac{xt}{m} \right.$$

Codificación C++:

```
double Leche(double m, double x,double t){
    return x/m*t;
};
```

Problema 2. *Los huevos.*

Si una tarecera parte de las aves que hay en la granja son gallinas, y la mitad de las gallinas ponen 1 huevo cada 3 días y la otra mitad 1 huevo cada 5 días, en un mes ¿Cuántos huevos producen? (1 mes = 30 días).

granja.h : Linea 5.

granja.cpp : Lineas 7-9.

Modelo Matemático: Se puede decucir que las gallinas que ponen huevos son solamente un sexto del total de aves, además la función debe devolver la parte entera porque una cantidad de huevos es un número natual.
Entonces:

$$Huevos : \mathbb{N} \to \mathbb{N}$$
$$(a) \mapsto \left\{ \lfloor \tfrac{a}{6} \left(\tfrac{30}{3} + \tfrac{30}{6} \right) \rfloor = \lfloor \tfrac{5}{2} a \rfloor \right.$$

Codificación C++:

```
int Huevos (int a){
    return (int)(5.0/2*a);
};
```

Problema 3. *La población de escorpiones.*

Si los escorpiones de la granja se venden a China, y hay escorpiones de tres diferentes tamaños, p - pequeños (con un peso de 20 gramos), m - medianos (con un peso 30 gramos) y g - grandes (con un peso de 50 gramos), ¿Cúantos kilos de escorpiones se pueden vender sin que decrezca la población a menos de dos tercios?

granja.h : Linea 6.

granja.cpp : Lineas 11-13.

Modelo Matemático: Unicamente se puede vender una tercera parte de los escorpiones entre pequeños, medianos y grandes; tambien se debe tener en cuenta las unidades de masa (convertir de gramos a kilogramos).
Entonces:

$$Escorpiones : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{R}$$
$$(p, m, g) \mapsto \left\{ \frac{1}{3} \left(p * \frac{2}{100} + m * \frac{3}{100} + g * \frac{5}{100} \right) \right.$$

Codificación C++:

```
double Escorpiones (int p, int m, int g){
    return (1/3.0)*(p*(2/100.0)+m*(3/100.0)+g*(5/100.0));
};
```

Problema 4. *La Reparación del Corral.*

Al granjero se le daño el corral y no sabe si volver a cercar el corral con madera, alambre de puas o poner reja de metal. Si va a cercar con madera debe poner 4 hileras de tablas, con varilla 8 hileras y con alambre solo 5 hileras, el quiere saber que es lo menos costoso para cercar si sabe que el alambre de puas vale p por metro, las tablas a q por metro y las varillas s por metro. Dado el tamaño del corral y los precios de los elementos, ¿cuál cercamiento es mas económico?

granja.h : Linea 7.

granja.cpp : Lineas 15-25.

Modelo Matemático: El tamaño del corral es irrelevante, ya que lo que importa es la cantidad de material utilizado por metro cuadrado.
Entonces:

$$Cerca : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{ASCII}^*$$

$$(p,q,s) \mapsto \begin{cases} Madera, & 4q \leq 5p \wedge 4q \leq 8s \\ Varilla, & 8s \leq 4q \wedge 8s \leq 5p \\ Alambre, & \text{En otro caso.} \end{cases}$$

Codificación C++:

```
char* Cerca (double p, double q, double s){
    if(q*4<=p*5&&q*4<=s*8){
        return "Madera";
    }else{
        if(s*8<=q*4&&s*8<=p*5){
            return "Varilla";
        }else{
            return "Alambre";
        };
    };
};
```

2. Numéricos.

Problema 5. *La potencia de un número.*
Función potencia de un entero elevado a un entero.

numericos.h : Linea 4.

numericos.cpp : Lineas 3-15.

Modelo Matemático:

$$\begin{aligned} Potencia : \mathbb{R} \times \mathbb{Z} &\to \mathbb{R} \\ (a, b) &\mapsto \Big\{ a^b \end{aligned}$$

Codificación C++:

```
double Potencia(double a, int b){
    if(b==0){
        return 1;
    }else{
        if(b<0){
            return 1.0/(Potencia(a,-b));
        }else{
            if(b>0){
                return a*Potencia(a,b-1);
            };
        };
    };
};
```

Problema 6. *Número que divide a otro.*
Una función que determine si un número es divisible por otro.

numericos.h : Linea 5.

numericos.cpp : Lineas 17-27.

Modelo Matemático:

$$\begin{aligned} Divisible : \mathbb{Z} \times \mathbb{Z} &\to \mathbb{B} \\ (a, b) &\mapsto \begin{cases} \mathbb{T}, & \frac{a}{b} \in \mathbb{Z} \\ \mathbb{F}, & \frac{a}{b} \notin \mathbb{Z} \vee b \neq 0 \end{cases} \end{aligned}$$

Codificación C++:

```
bool Divisible(int a, int b){
    if(b==0){
        return false;
    }else{
        if(a%b==0){
            return true;
        }else{
            return false;
        };
    };
};
```

Problema 7. *Número primo.*
Determinar si un número es primo.

numericos.h : Linea 6.

numericos.cpp : Lineas 29-47.

Modelo Matemático:

$$Primo : \mathbb{Z}^+ \to \mathbb{B}$$
$$(a) \mapsto \begin{cases} \mathbb{T}, & a = 2 \vee \forall n \in (\mathbb{Z}^+ - \{1, a\}) : \frac{a}{n} \notin \mathbb{Z} \\ \mathbb{F}, & a = 1 \vee \exists n \in (\mathbb{Z}^+ - \{1, a\}) : \frac{a}{n} \in \mathbb{Z} \end{cases}$$

Codificación C++:

```
bool Primo (int a){
    int n = a-1;
    if(a==2){
        return true;
    }else{
        if(a==1){
            return false;
        }else{
            while(n>1){
                if(a%n==0) {
                    return false;
                }else{
                    n--;
                };
```

```
            };
            return true;
        };
    };
};
```

Problema 8. *Primos relativos.*
Dados dos naturales, determinar si son primos relativos.

numericos.h : Linea 7.

numericos.cpp : Lineas 49-66.

Modelo Matemático:

$$PrimosRelativos : \mathbb{N} \times \mathbb{N} \to \mathbb{B}$$
$$(a,b) \mapsto \begin{cases} \mathbb{T}, & \forall n \in (\mathbb{Z}^+ - \{1\}) : \left\{\frac{a}{n}, \frac{b}{n}\right\} \not\subset \mathbb{Z} \\ \mathbb{F}, & \exists n \in (\mathbb{Z}^+ - \{1\}) : \left\{\frac{a}{n}, \frac{b}{n}\right\} \subset \mathbb{Z} \end{cases}$$

Codificación C++:

```
bool PrimosRelativos(int a, int b){
    int n = a;
    int m = b;
    while(n>1&&m>1){
        if((a%n==0&&b%n==0)||(a%m==0&&b%m==0)){
            return false;
        }else{
            if(n>1){
                n--;
            }else{
                if(m>1){
                    m--;
                };
            };
        };
    };
    return true;
};
```

Problema 9. *Multiplo de la suma.*
Determinar si un número es múltiplo de la suma de otros dos números.

numericos.h : Linea 8.

numericos.cpp : Lineas 68-77.

Modelo Matemático:

$$MultiploSuma : \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \to \mathbb{B}$$
$$(a,b,c) \mapsto \begin{cases} \mathbb{T}, & \frac{a}{b+c} \in \mathbb{Z}, \ \ b+c \neq 0 \\ \mathbb{F}, & \frac{a}{b+c} \notin \mathbb{Z}, \ \ b+c \neq 0 \end{cases}$$

Codificación C++:

```
bool MultiploSuma (int a, int b, int c){
    if(b+c==0) {
        return false;
    }else{
        if(a%(b+c)==0){
            return true;
        };
        return false;
    };
};
```

Problema 10. *Evaluación de un polinomio.*

Dados los coeficientes de un polinomio de grado dos, evaluar el polinomio en un punto dado.

numericos.h : Linea 9.

numericos.cpp : Lineas 79-81.

Modelo Matemático:

$$EvaluarPolinomio : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$
$$(a,b,c,x) \mapsto \Big\{ ax^2 + bx + c$$

Codificación C++:

```
double EvaluarPolinomio (double a, double b, double c, double x){
    return a*x*x+b*x+c;
};
```

Problema 11. *El coeficiente lineal de la derivada.*

Dados los coeficientes de un polinomio de grado dos, calcular coeficiente lineal de la derivada.

numericos.h : Linea 10.

numericos.cpp : Lineas 83-85.

Modelo Matemático:

$$\begin{aligned} Coeficiente Lineal : \mathbb{R} \times \mathbb{R} \times \mathbb{R} &\to \mathbb{R} \\ (a, b, c) &\mapsto \Big\{ 2a \end{aligned}$$

Codificación C++:

```
double CoeficienteLineal (double a, double b, double c){
    return 2*a;
};
```

Problema 12. *La derivada en un punto.*

Dados los coeficientes de un polinomio de grado dos, calcular la derivada en un punto dado.

numericos.h : Linea 11.

numericos.cpp : Lineas 87-89.

Modelo Matemático:

$$\begin{aligned} Derivada : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} &\to \mathbb{R} \\ (a, b, c, x) &\mapsto \Big\{ 2ax + b \end{aligned}$$

Codificación C++:

```
double Derivada (double a, double b, double c, double x){
    return 2*a*x+b;
};
```

Problema 13. *La Serie de Fibonacci.*

Dado un natural, determinar si es un número de fibonacci o no.

numericos.h : Linea 12.

numericos.cpp : Lineas 91-107.

Modelo Matemático: Los números que conforman la sucesión de Fibonacci se definen como la suma de los dos anteriores en dicha sucesión de la siguiente manera:

$$\begin{aligned} F_n = \ & F_{n-1} + F_{n-2} \\ & F_1 = 0 \\ & F_2 = 1 \end{aligned}$$

Entonces:

$$Fibonacci : \mathbb{N} \to \mathbb{B}$$

$$(n) \mapsto \begin{cases} \mathbb{T}, & \exists x \in \mathbb{N} : F_x = n \\ \mathbb{F}, & \text{En otro caso} \\ & F_1 = 0 \\ & F_2 = 1 \\ & F_x = F_{x-1} + F_{x-2} \end{cases}$$

Codificación C++:

```
bool Fibonacci(int n){
    if(n==1||n==0){
        return true;
    };
    int x = 1;
    int y = 1;
    int z = 0;
    for(int i =0;i<=n;i++){
        z=x+y;
        x=y;
        y=z;
        if(n==z){
            return true;
        };
    };
    return false;
};
```

3. Geométricos.

Problema 14. *Rectas en el plano.*

Dados la pendiente y el punto de corte de dos rectas, determinar si son paralelas, perpendiculares o ninguna de las anteriores.

geometricos.h : Linea 4.

geometricos.cpp : Lineas 7-17.

Modelo Matemático:

$$Rectas : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{ASCII}^*$$

$$(m_1, m_2, b_1, b_2) \mapsto \begin{cases} Paralelas, & m_1 = m_2 \ \wedge \ b_1 \neq b_2 \\ Perpenticulares, & -\frac{1}{m_1} = m_2 \\ Ninguna, & \text{En otro caso.} \end{cases}$$

Codificación C++:

```
char* Rectas(double m1, double m2, double b1, double b2){
    if(m1==m2&&b1!=b2){
        return "Paralelas";
    }else{
        if(m1*m2==-1){
            return "Perpenticulares";
        }else{
            return "Ninguna";
        };
    };
};
```

Problema 15. *Intersección de dos rectas en el plano.*

Dados la pendiente y el punto de corte de dos rectas, determinar el punto de intersección.

geometricos.h : Linea 5.

geometricos.cpp : Lineas 19-24.

Modelo Matemático:

$$Interseccion : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^*$$

$$(m_1, m_2, b_1, b_2) \mapsto \left\{ \left(\frac{b_2 - b_1}{m_1 - m_2}, m_1 * \frac{b_2 - b_1}{m_1 - m_2} + b_1 \right) \right.$$

Codificación C++:

```
double* Corte(double m1, double m2, double b1, double b2){
    double* r = new double [2];
    r[0]=(b2-b1)/(m1-m2);
    r[1]=m1*((b2-b1)/(m1-m2))+b1;
    return r;
};
```

Nota: En los proximos ejercicios se usara la función raíz cuadrada y la función valor absoluto con los siguientes modelos matemáticos y codificaciones c++.

Problema 16. *Valor Absoluto.*

Reales.h : Linea 6.

Reales.cpp : Lineas 14-22.

Modelo Matemático:

$$ValorAbsoluto : \mathbb{R} \to \mathbb{R}$$
$$(x) \mapsto \begin{cases} x, & x \geq 0 \\ -x, & x < 0 \end{cases}$$

Codificación C++:

```
double ValorAbsoluto(double x){
    if(x<0){
        return -x;
    }else{
        if(x>=0){
            return x;
        };
    };
};
```

Problema 17. *Raíz cuadrada de un número.*

Reales.h : Linea 5.

Reales.cpp : Lineas 4-12.

Modelo Matemático:

$$Raiz2 : \mathbb{R} \to \mathbb{R}$$
$$(x) \mapsto \left\{ \sqrt{x} \right.$$

Codificación C++:

```
double Raiz2(double x){
    double Xo;
    double Xi = x;
    do{
        Xo = Xi;
        Xi = 0.5 * (Xo + x / Xo);
    }while(ValorAbsoluto(Xo - Xi) >= 1e-20);
    return 0.5 * (Xi + x / Xi);
};
```

Problema 18. *Área de un triángulo Circunscrito a una circunferencia.*

geometricos.h : Linea 6.

geometricos.cpp : Lineas 26-28.

Modelo Matemático:

$$AreaTrianguloCircuns : \mathbb{R} \to \mathbb{R}$$
$$(r) \mapsto \left\{ 3\sqrt{3}r^2 \right.$$

Codificación C++:

```
double AreaTrianguloCircuns (double r){
    return 3*Raiz2(3)*Potencia(r,2);
};
```

Problema 19. *Perímetro de un triángulo Circunscrito a una circunferencia.*

geometricos.h : Linea 7.

geometricos.cpp : Lineas 30-32.

Modelo Matemático:

$$PerimetroTrianguloCircuns : \mathbb{R} \to \mathbb{R}$$
$$(r) \mapsto \left\{ 6\sqrt{3}r \right.$$

Codificación C++:

```
double PerímetroTrianguloCircuns (double r){
    return 6*Raiz2(3)*r;
};
```

Problema 20. *Área de un triángulo inscrito en una circunferencia.*

geometricos.h : Linea 8.

geometricos.cpp : Lineas 34-36.

Modelo Matemático:

$$AreaTrianguloInsc : \mathbb{R} \to \mathbb{R}$$
$$(r) \mapsto \left\{ \frac{3\sqrt{3}r^2}{4} \right.$$

Codificación C++:

```
double AreaTrianguloInsc (double r){
    return 3*Raiz2(3)/4*Potencia(r,2);
};
```

Problema 21. *Perímetro de un triángulo inscrito en una circunferencia.*

geometricos.h : Linea 9.

geometricos.cpp : Lineas 38-40.

Modelo Matemático:

$$PerimetroTrianguloInsc : \mathbb{R} \to \mathbb{R}$$
$$(r) \mapsto \left\{ 3\sqrt{3}r \right.$$

Codificación C++:

```
double PerimetroTrianguloInsc (double r){
    return r*Raiz2(3)*3;
};
```

Problema 22. *Área de un cuadrado circunscrito a una circunferencia.*

geometricos.h : Linea 10.

geometricos.cpp : Lineas 42-44.

Modelo Matemático:

$$AreaCuadradoCircuns : \mathbb{R} \to \mathbb{R}$$
$$(r) \mapsto \Big\{ 4r^2$$

Codificación C++:

```
double AreaCuadradoCircuns (double r){
    return Potencia(2*r,2);
};
```

Problema 23. *Perímetro de un cuadrado circunscrito a una circunferencia.*

geometricos.h : Linea 11.

geometricos.cpp : Lineas 46-48.

Modelo Matemático:

$$PerimetroCuadradoCircuns : \mathbb{R} \to \mathbb{R}$$
$$(r) \mapsto \Big\{ 8r$$

Codificación C++:

```
double PerimetroCuadradoCircuns (double r){
    return 8*r;
};
```

Problema 24. *Área de un cuadrado inscrito en una circunferencia.*

geometricos.h : Linea 12.

geometricos.cpp : Lineas 50-52.

Modelo Matemático:

$$AreaCuadradoInsc : \mathbb{R} \to \mathbb{R}$$
$$(r) \mapsto \Big\{ 2r^2$$

Codificación C++:

```
double AreaCuadradoInsc (double r){
    return 2*Potencia(r,2);
};
```

Problema 25. *Perímetro de un cuadrado inscrito en una circunferencia.*

geometricos.h : Linea 13.

geometricos.cpp : Lineas 54-56.

Modelo Matemático:

$$PerimetroCuadradoInsc : \mathbb{R} \to \mathbb{R}$$
$$(r) \mapsto \Big\{ 4\sqrt{2}r$$

Codificación C++:

```
double PerimetroCuadradoInsc (double r){
    return 4*Raiz2(2)*r;
};
```

Nota: En los proximos ejercicios se usaran las tres funciones trigonométricas básicas y la función factorial con los siguientes modelos matemáticos y codificaciones c++

Problema 26. *Factorial.*

Enteros.h : Linea 4.

Enteros.cpp : Lineas 3-9.

Modelos Matemáticos:

$$Factorial : \mathbb{N} \to \mathbb{N}$$
$$(x) \mapsto \Big\{ x!$$

Codificación C++:

```
int Factorial(int x){
    if(x==0){
        return 1;
    }else{
        return x*Factorial(x-1);
    };
};
```

Problema 27. *Seno de un angulo en radianes.*

Reales.h : Linea 7.

Reales.cpp : Lineas 24-31.

Modelos Matemáticos:

$$\begin{aligned} Seno : \mathbb{R} &\to \mathbb{R} \\ (x) &\mapsto \left\{ \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} \right. \end{aligned}$$

Codificación C++:

```
double Seno(double x){
    double a =0;
    for(int i=0;i=<12;i++){//Se toma 12 como valor finito.
        a+=((Potencia(-1,i)*Potencia(x,2*i+1))/Factorial(2*i+1));
    };
    return a;
};
```

Problema 28. *Coseno de un angulo en radianes.*

Reales.h : Linea 8.

Reales.cpp : Lineas 33-39.

Modelos Matemáticos:

$$\begin{aligned} Coseno : \mathbb{R} &\to \mathbb{R} \\ (x) &\mapsto \left\{ \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} \right. \end{aligned}$$

Codificación C++:

```
double Coseno(double x){
    double a =0;
    for(int i=0;i=<12;i++){//Se toma 12 como valor finito.
        a+=((Potencia(-1,i)*Potencia(x,2*i))/Factorial(2*i));
    };
    return a;
};
```

Problema 29. *Tangente de un angulo en radianes.*

Reales.h : Linea 9.

Reales.cpp : Lineas 41-43.

Modelos Matemáticos:

$$Tangente : \mathbb{R} \to \mathbb{R}$$
$$(x) \mapsto \left\{ \frac{Seno(x)}{Coseno(x)} \right.$$

Codificación C++:

```
double Tangente(double x){
    return ((Seno(x))/(Coseno(x)));
};
```

Problema 30. *Área de un pentágono circunscrito a una circunferencia.*

geometricos.h : Linea 14.

geometricos.cpp : Lineas 58-60.

Modelo Matemático:

$$AreaPentagonoCircuns : \mathbb{R} \to \mathbb{R}$$
$$(r) \mapsto \left\{ 5 * Tangente(\tfrac{\pi}{5}) * r^2 \right.$$

Codificación C++:

```
double const Pi = 3.14159265359;

double AreaPentagonoCircuns(double r){
    return 5*Tangente(Pi/5)*Potencia(r,2);
};
```

Problema 31. *Perimetro de un pentágono circunscrito a una circunferencia.*

geometricos.h : Linea 15.

geometricos.cpp : Lineas 62-64.

Modelo Matemático:

$$PerimetroPentagonoCircuns : \mathbb{R} \to \mathbb{R}$$
$$(r) \mapsto \Big\{ 10 * Tangente(\tfrac{\pi}{5}) * r$$

Codificación C++:

```
double const Pi = 3.14159265359;

double PerimetroPentagonoCircuns(double r){
    return 10*Tangente(Pi/5)*r;
};
```

Problema 32. *Área de un pentágono inscrito en una circunferencia.*

geometricos.h : Linea 16.

geometricos.cpp : Lineas 66-68.

Modelo Matemático:

$$AreaPentagonoInsc : \mathbb{R} \to \mathbb{R}$$
$$(r) \mapsto \Big\{ 5 * Seno(\tfrac{\pi}{5}) * Coseno(\tfrac{\pi}{5}) * r^2$$

Codificación C++:

```
double const Pi = 3.14159265359;

double AreaPentagonoInsc (double r){
    return 5*Seno(Pi/5)*Coseno(Pi/5)*Potencia(r,2);
};
```

Problema 33. *Perímetro de un pentágono inscrito en una circunferencia.*

geometricos.h : Linea 17.

geometricos.cpp : Lineas 70-72.

Modelo Matemático:

$$PerimetroPentagonoInsc : \mathbb{R} \to \mathbb{R}$$
$$(r) \mapsto \Big\{ 10 * Seno(\tfrac{\pi}{5}) * r$$

Codificación C++:

```
double const Pi = 3.14159265359;

double PerimetroPentagonoInsc(double r){
    return 10*Seno(Pi/5)*r;
};
```

Problema 34. *Área de un hexágono circunscrito en una circunferencia.*

geometricos.h : Linea 18.

geometricos.cpp : Lineas 74-76.

Modelo Matemático:

$$AreaHexagonoCircuns : \mathbb{R} \to \mathbb{R}$$
$$(r) \mapsto \Big\{ 6 * Tangente(\tfrac{\pi}{6}) * r^2$$

Codificación C++:

```
double const Pi = 3.14159265359;

double AreaHexagonoCircuns(double r){
    return 6*Tangente(Pi/6)*Potencia(r,2);
};
```

Problema 35. *Perímetro de un hexágono circunscrito en una circunferencia.*

geometricos.h : Linea 19.

geometricos.cpp : Lineas 78-80.

Modelo Matemático:

$$PerimetroHexagonoCircuns : \mathbb{R} \to \mathbb{R}$$
$$(r) \mapsto \Big\{ 12 * Tangente(\tfrac{\pi}{6}) * r$$

Codificación C++:

```
double const Pi = 3.14159265359;

double PerimetroHexagonoCircuns(double r){
    return 12*Tangente(Pi/6)*r;
};
```

Problema 36. *Área de un hexágono inscrito en una circunferencia.*

geometricos.h : Linea 20.

geometricos.cpp : Lineas 82-84.

Modelo Matemático:

$$AreaHexagonoInsc : \mathbb{R} \to \mathbb{R}$$
$$(r) \mapsto \left\{ \frac{3\sqrt{3}}{2} r \right.$$

Codificación C++:

```
double AreaHexagonoInsc(double r){
    return 3*Raiz2(3)/2*Potencia(r,2);
};
```

Problema 37. *Perímetro de un hexágono inscrito en una circunferencia.*

geometricos.h : Linea 21.

geometricos.cpp : Lineas 86-88.

Modelo Matemático:

$$PerimetroHexagonoInsc : \mathbb{R} \to \mathbb{R}$$
$$(r) \mapsto \Big\{ 6r$$

Codificación C++:

```
double PerimetroHexagonoInsc(double r){
    return 6*r;
};
```

Problema 38. *La Telaraña.*

Si una araña utiliza un patrón hexagonal para su telaraña, y cada hexágono está separado del otro por 1 cm, y la araña quiere hacer una telaraña de πr^2, ¿Qué cantidad de telaraña requiere la araña?

geometricos.h : Linea 22.

geometricos.cpp : Lineas 90-96.

Modelo Matemático:

$$Telarana : \mathbb{R} \to \mathbb{R}$$
$$(r) \mapsto \left\{ \sum_{i=1}^{\lfloor r \rfloor} 6i + 6r \right.$$

Codificación C++:

```
double Telarana (double r){
    double x = 6*r;
    for(int i = 1;i<=(int)r;i++){
        x+=6*i;
    };
    return x;
};
```

4. Otros.

Problema 39. *Las hojas de los árboles.*

Si en la UN están podando árboles y cada rama tiene p hojas, y a cada árbol le quitaron k ramas, ¿Cuántos árboles se deben podar para obtener t hojas?

otros.h : Linea 4.

otros.cpp : Lineas 4-14.

Modelo Matemático: La relación entre árboles, ramas y hojas es directamente proporcional; para ello se establecen las siguientes variables:

$p :=$ Cantidad de hojas que tiene una rama.

$k :=$ Cantidad de ramas que tiene un árbol.

$t :=$ Cantidad de hojas en total.

Entonces:

$$Arbol : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$$
$$(k,p,t) \mapsto \begin{cases} 0, & t = 0 \vee p = 0 \vee k = 0 \\ \frac{t}{pk}, & \frac{t}{pk} \in \mathbb{Z} \\ \lfloor \frac{t}{pk} \rfloor + 1, & \text{En otro caso} \end{cases}$$

Codificación C++:

```
int Arbol(int k,int p,int t){
    if(p==0||k==0||t==0){
        return 0;
    }else{
        if((t%(p*k))==0){
            return (int)(t/(p*k));
        }else{
            return ((int)(t/(p*k)))+1;
        };
    };
};
```

Problema 40. *Interes simple.*

Si un amigo, no tan amigo, me presta k pesos a i pesos de interés diario, ¿Cuánto le pagaré en una semana si el interés es simple?

otros.h : Linea 5.

otros.cpp : Lineas 16-18.

Modelo Matemático: En el interes simple el aumento se da de manera lineal.

Entonces:

$$InteresSimple : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$
$$(k,i) \mapsto \Big\{ 7i + k$$

Codificación C++:

```
double InteresSimple(double k, double i){
    return k+7*i;
};
```

Problema 41. *Interes Compuesto.*

Si un amigo, no tan amigo, me presta k pesos a i pesos de interés diario, ¿Cuánto le pagaré en una semana si el interés es compuesto?

otros.h : Linea 6.

otros.cpp : Lineas 20-22.

Modelo Matemático: En el interes simple el aumento se da de manera exponencial. Entonces:

$$\begin{aligned} InteresCompuesto : \mathbb{R} \times \mathbb{R} &\to \mathbb{R} \\ (k, i) &\mapsto \left\{ k + \left(1 + \frac{i}{k}\right)^7 \right. \end{aligned}$$

Codificación C++:

```
double InteresCompuesto(double k, double i){
    return k+Potencia(1+i/k,7);
};
```

Problema 42. *Lego con fichas rojas y azules.*

Un niño se la pasó jugando con fichas de lego, tenia dos tipos de fichas de lego, fichas de cuadros de 1x1 (rojas) y fichas azules de 2x1 (azules), y le dieron una base de 1xn cuadrito, ¿Cuántas formas diferentes puede construir con esa hilera?

otros.h : Linea 7.

otros.cpp : Lineas 24-31.

Modelo Matemático: La cantidad de posibilidades de llenar una base de lego crecen siguiendo la serie de Fibonacci. Entonces:

$$\begin{aligned} Lego1 : \mathbb{N} &\to \mathbb{N} \\ (n) &\mapsto \begin{cases} Lego1(n-1) + Lego1(n-2), & n > 1 \\ 1, & n = 0 \\ 1, & n = 1 \end{cases} \end{aligned}$$

Codificación C++:

```
double Lego1 (int n){
    if(n==0||n==1){
        return 1;
```

```
    }else{
        return Lego1(n-1)+Lego1(n-2);

    };
};
```

Problema 43. *Lego con fichas rojas, azules y amarillas.*

Un niño se la pasó jugando con fichas de lego, tenia tres tipos de fichas de lego, fichas de cuadros de 1x1 (rojas), fichas azules de 2x1 (azules)y fichas de cuadros 3x1 (amarillas) y le dieron una base de 1xn cuadrito, ¿Cuántas formas diferentes puede construir con esa hilera?

otros.h : Linea 8.

otros.cpp : Lineas 33-43.

Modelo Matemático: Este modelo matemático se basa en una modificación de la sucesión de Fibonacci, en dicha sucesión el elemento F_n es igual a la suma de los dos elementos anteriores, pero se modifica para tomar la suma de los tres anteriores, la nueva sucesión es denominada *"Tribonacci"*, la serie seria representada de la siguiente forma:

$$\begin{aligned} T_n = \ & T_{n-1} + T_{n-2} + T_{n-3} \\ & T_0 = 0 \\ & T_1 = 1 \\ & T_2 = 1 \end{aligned}$$

Pero en este caso se modifica la sucesión con la suma de los tres anteriores, también llamada sucesión de *"Tribonacci"*
Entonces:

$Lego2 : \mathbb{N} \to \mathbb{N}$

$$(n) \mapsto \begin{cases} Lego2(n-1) + Lego2(n-2) + Lego2(n-3), & n > 3 \\ 1, & n = 0 \vee n = 1 \\ 2, & n = 2 \end{cases}$$

Codificación C++:

```
double Lego2 (int n){
    if(n==0||n==1){
        return 1;
    }else{
        if(n==2){
            return 2;
        }else{
            return Lego2(n-1)+Lego2(n-2)+Lego2(n-3);
        };
    };
};
```

5. Problemas de Arreglos.

Problema 44. *Criba de Eratostenes.*

Implementar la criba de Eratostenes para calcular los números primos en el rango 1 a n, donde n es un número natural dado por el usuario.

probarreblos.h : Linea 4.

probarreblos.cpp : Lineas 7-30.

Modelo Matemático: Para este problema se utilizan las funciones primosiguiente y contarprimos definidas mas adelante.
Entonces:

$$CribaEratostenes : \mathbb{N} \to \mathbb{N}^*$$
$$(n) \mapsto \left\{ \cup_x : x \in \mathbb{N} \wedge 1 < x \leq n \wedge primo(x) \right.$$

Codificación C++:

```
int* CribaEratostenes (int n){
    int* criba = new int[n+1];
    criba[0]=0;
    criba[1]=0;
    for(int i=2;i<=n;i++){
        criba[i]=1;
    };
    int x = 2;
    while(x<=(int)(Raiz2((double)n))){
        for(int i = 2;i*x<=n;i++){
            criba[x*i]=0;
        };
        x=primosiguiente(x);
    };
    int* X = new int[contarprimos(criba,n+1)];
    int c =0;
    for(int i=2;i<=n;i++){
        if(criba[i]==1){
          X[c]=i;
          c++;
        };
    };
    return X;
};
```

Problema 45. *Primo siguiente.*

probarreblos.h : Linea 5.

probarreblos.cpp : Lineas 32-40.

Modelo Matemático:

$$primosiguiente : \mathbb{N} \to \mathbb{N}$$
$$(x) \mapsto \left\{ x+i, \quad primo(x+i); \forall_{i=1}^{\infty} \right.$$

Codificación C++:

```
int primosiguiente(int x){
    x++;
  while(true){
    if(Primo(x)){
        return x;
    };
    x++;
  };
};
```

Problema 46. *Contar Primos.*

probarreblos.h : Linea 6.

probarreblos.cpp : Lineas 42-48.

Modelo Matemático:

$$contarprimos : \mathbb{Z}^* \times \mathbb{N} \to \mathbb{N}$$
$$(Z, n) \mapsto \left\{ \sum_{i=1}^{n} Z_i \right.$$

Codificación C++:

```
int contarprimos(int* Z,int n){
    int x=0;
    for(int i = 0; i<n;i++){
        x+=Z[i];
    };
    return x;
};
```

Problema 47. *Suma de un arreglo(Enteros).*

Desarrollar un algoritmo que calcule la suma de los elementos de un arreglo de números enteros.

probarreblos.h : Linea 7.

probarreblos.cpp : Lineas 50-56.

Modelo Matemático:

$$SumaArrEnteros : \mathbb{Z}^* \times \mathbb{N} \to \mathbb{Z}$$
$$(Z, n) \mapsto \left\{ \sum_{i=1}^{n} Z_i \right.$$

Codificación C++:

```
int SumaArrEnteros (int* Z,int n){
    int x = 0;
    for(int i=0;i<n;i++){
        x+=Z[i];
    };
    return x;
};
```

Problema 48. *Suma de un arreglo(Reales).*

Desarrollar un algoritmo que calcule la suma de los elementos de un arreglo de números reales.

probarreblos.h : Linea 8.

probarreblos.cpp : Lineas 58-64.

Modelo Matemático:

$$SumaArrReales : \mathbb{R}^* \times \mathbb{N} \to \mathbb{R}$$
$$(R, n) \mapsto \left\{ \sum_{i=1}^{n} R_i \right.$$

Codificación C++:

```
double SumaArrReales (double* R,int n){
    double x = 0;
    for(int i=0;i<n;i++){
        x+=R[i];
    };
    return x;
};
```

Problema 49. *Promedio de los elementos un arreglo(Enteros).*
Desarrollar un algoritmo que calcule el promedio de un arreglo de enteros.

probarreblos.h : Linea 9.

probarreblos.cpp : Lineas 66-68.

Modelo Matemático:

$$PromedioArrEnteros : \mathbb{Z}^* \times \mathbb{N} \to \mathbb{R}$$
$$(Z, n) \mapsto \left\{ \frac{SumaArrEnteros(Z, n)}{n} \right.$$

Codificación C++:

```
double PromedioArrEnteros(int* Z,int n){
    return SumaArrEnteros(Z,n)/n;
};
```

Problema 50. *Promedio de los elementos un arreglo(Reales).*
Desarrollar un algoritmo que calcule el promedio de un arreglo de reales.

probarreblos.h : Linea 10.

probarreblos.cpp : Lineas 70-72.

Modelo Matemático:

$$PromedioArrReales : \mathbb{R}^* \times \mathbb{N} \to \mathbb{R}$$
$$(R, n) \mapsto \left\{ \frac{SumaArrReales(R, n)}{n} \right.$$

Codificación C++:

```
double PromedioArrReales(double* R,int n){
    return SumaArrReales(R,n)/n;
};
```

Problema 51. *Producto de dos arreglos (Enteros).*

Desarrollar un algoritmo que calcule el producto de dos arreglos de números enteros de igual tamaño. Sean $v = (v_1, v_2, ..., v_n)$ y $w = (w_1, w_2, ..., w_n)$ dos arreglos, el producto de v y w (notado $v.w$) es el número: $v_1 * w_1 + v_2 * w_2 + ... + v_n * w_n$

probarreblos.h : Linea 11.

probarreblos.cpp : Lineas 74-84.

Modelo Matemático:

$$ProductoArrEnteros : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{N} \to \mathbb{Z}$$
$$(Z, S, n) \mapsto \left\{ \sum_{i=1}^{n} Z_i S_i \right.$$

Codificación C++:

```
int ProductoArrEnteros(int* Z,int* S,int n){
    int* Q= new int[n];
    for(int i=0;i<n;i++){
        Q[i]=S[i]*Z[i];
    };
    int x = 0;
    for(int j=0;j<n;j++){
        x +=Q[j];
    };
    return x;
};
```

Problema 52. *Producto de dos arreglos (Reales).*

Desarrollar un algoritmo que calcule el producto de dos arreglos de números reales de igual tamaño. Sean $v = (v_1, v_2, ..., v_n)$ y $w = (w_1, w_2, ..., w_n)$ dos arreglos, el producto de v y w (notado $v.w$) es el número: $v_1 * w_1 + v_2 * w_2 + ... + v_n * w_n$

probarreblos.h : Linea 12.

probarreblos.cpp : Lineas 86-96.

Modelo Matemático:

$$ProductoArrReales : \mathbb{R}^* \times \mathbb{R}^* \times \mathbb{N} \to \mathbb{R}$$
$$(R, S, n) \mapsto \left\{ \sum_{i=1}^{n} R_i S_i \right.$$

Codificación C++:

```
double ProductoArrReales(double* R,double* S,int n){
    double* Q= new double[n];
    for(int i=0;i<n;i++){
        Q[i]=S[i]*R[i];
    };
    double x = 0;
    for(int j = 0;j<n;j++){
        x+=Q[j];
    };
    return x;
};
```

Problema 53. *Mínimo de un arreglo (Enteros).*

Desarrollar un algoritmo que calcule el mínimo de un arreglo de números enteros.

probarreblos.h : Linea 13.

probarreblos.cpp : Lineas 96-106.

Modelo Matemático:

$$MinimoArrEnteros : \mathbb{Z}^* \times \mathbb{N} \to \mathbb{Z}$$
$$(Z, n) \mapsto \left\{ Z_i : Z_i \leq Z_a \forall_{a=1}^{n} \right.$$

Codificación C++:

```
int MinimoArrEnteros(int* Z,int n){
    int x = Z[0];
    for(int i=1;i<n;i++){
        if(Z[i]<x){
            x = Z[i];
        };
    };
    return x;
};
```

Problema 54. *Mínimo de un arreglo (Reales).*

Desarrollar un algoritmo que calcule el mínimo de un arreglo de números reales.

probarreblos.h : Linea 14.

probarreblos.cpp : Lineas 108-116.

Modelo Matemático:

$$MinimoArrReales : \mathbb{R}^* \times \mathbb{N} \to \mathbb{R}$$
$$(R, n) \mapsto \Big\{ R_i : R_i \leq R_a \forall_{a=1}^{n}$$

Codificación C++:

```
double MinimoArrReales(double* R,int n){
    double x = R[0];
    for(int i=1;i<n;i++){
        if(R[i]<x){
            x = R[i];
        };
    };
    return x;
};
```

Problema 55. *Máximo de un arreglo (Enteros).*

Desarrollar un algoritmo que calcule el máximo de un arreglo de números enteros.

probarreblos.h : Linea 15.

probarreblos.cpp : Lineas 118-126.

Modelo Matemático:

$$MaximoArrEnteros : \mathbb{Z}^* \times \mathbb{N} \to \mathbb{Z}$$
$$(Z, n) \mapsto \Big\{ Z_i : Z_i \geq Z_a \forall_{a=1}^{n}$$

Codificación C++:

```
int MaximoArrEnteros(int* Z,int n){
    int x = Z[0];
    for(int i=1;i<n;i++){
        if(Z[i]>x){
            x = Z[i];
        };
    };
    return x;
};
```

Problema 56. *Máximo de un arreglo (Reales).*

Desarrollar un algoritmo que calcule el máximo de un arreglo de números reales.

probarreblos.h : Linea 16.

probarreblos.cpp : Lineas 128-136.

Modelo Matemático:

$$MaximoArrReales : \mathbb{R}^* \times \mathbb{N} \to \mathbb{R}$$
$$(R, n) \mapsto \left\{ R_i : R_i \geq R_a \forall_{a=1}^{n} \right.$$

Codificación C++:

```
double MaximoArrReales(double* R,int n){
    double x = R[0];
    for(int i=1;i<n;i++){
        if(R[i]>x){
            x = R[i];
        };
    };
    return x;
};
```

Problema 57. *Producto dirécto de un arreglo (Enteros).*

Desarrollar un algoritmo que calcule el producto directo de dos arreglos de enteros de igual tamaño. Sean $v = (v_1, v_2, ..., v_n)$ y $w = (w_1, w_2, ..., w_n)$ dos arreglos, el producto directo de v y w (notado $v * w$) es el vector: $[v_1 * w_1, v_2 * w_2, ..., v_n * w_n]$

probarreblos.h : Linea 17.

probarreblos.cpp : Lineas 138-144.

Modelo Matemático:

$$ProductoDArrEnteros : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{N} \to \mathbb{Z}^*$$
$$(Z, S, n) \mapsto \left\{ \cup_{x=1}^{n} Z_x S_x \right.$$

Codificación C++:

```
int* ProductoDArrEnteros(int* Z,int* S,int n){
    int* Q= new int[n];
```

```
    for(int i=0;i<n;i++){
        Q[i]=S[i]*Z[i];
    };
    return Q;
};
```

Problema 58. *Producto dirécto de un arreglo (Reales).*

Desarrollar un algoritmo que calcule el producto directo de dos arreglos de reales de igual tamaño. Sean $v = (v_1, v_2, ..., v_n)$ y $w = (w_1, w_2, ..., w_n)$ dos arreglos, el producto directo de v y w (notado $v * w$) es el vector: $[v_1 * w_1, v_2 * w_2, ..., v_n * w_n]$

probarreblos.h : Linea 18.

probarreblos.cpp : Lineas 146-152.

Modelo Matemático:

$$ProductoDArrReales : \mathbb{R}^* \times \mathbb{R}^* \times \mathbb{N} \to \mathbb{R}^*$$
$$(R, S, n) \mapsto \left\{ \cup_{x=1}^{n} R_x S_x \right.$$

Codificación C++:

```
double* ProductoDArrReales(double* R,double* S,int n){
    double* Q= new double[n];
    for(int i=0;i<n;i++){
        Q[i]=S[i]*R[i];
    };
    return Q;
};
```

Problema 59. *Mediana de un arreglo (Enteros).*

Desarrollar un algoritmo que determine la mediana de un arreglo de enteros. La mediana es el número que queda en la mitad del arreglo después de ser ordenado.

probarreblos.h : Linea 19.

probarreblos.cpp : Lineas 154-170.

Modelo Matemático:

$$MedianaArrEnteros : \mathbb{Z}^* \times \mathbb{N} \to \mathbb{R}$$
$$(Z, n) \mapsto \begin{cases} Z_{\frac{n+1}{2}}, & \frac{n}{2} \notin \mathbb{Z} \\ \frac{Z_{\frac{n}{2}} + Z_{\frac{n}{2}+1}}{2}, & \text{En otro caso} \end{cases}$$

Codificación C++:

```
double MedianaArrEnteros(int* Z,int n){
    int c;
    for(int j = 0;j<n-1;j++){
        for(int i = 0; i<n-1;i++){
            if(Z[i]>Z[i+1]){
                c=Z[i+1];
                Z[i+1]=Z[i];
                Z[i]=c;
            };
        };
    };
    if(n%2==0){
        return (double)(Z[n/2]+Z[(n/2)-1])/2;
    }else{
        return (double) Z[(n-1)/2];
    };
};
```

Problema 60. *Mediana de un arreglo (Reales).*

Desarrollar un algoritmo que determine la mediana de un arreglo de reales. La mediana es el número que queda en la mitad del arreglo después de ser ordenado.

probarreblos.h : Linea 20.

probarreblos.cpp : Lineas 172-188.

Modelo Matemático:

$$MedianaArrReales : \mathbb{R}^* \times \mathbb{N} \to \mathbb{R}$$

$$(R, n) \mapsto \begin{cases} R_{\frac{n+1}{2}}, & \frac{n}{2} \notin \mathbb{Z} \\ \dfrac{R_{\frac{n}{2}} + R_{\frac{n}{2}+1}}{2}, & \text{En otro caso} \end{cases}$$

El arreglo R debió ser anteriormente ordenado para que la función sirva.

Codificación C++:

```
double MedianaArrReales(double* R,int n){
    double c;
    for(int j = 0;j<n-1;j++){
        for(int i = 0; i<n-1;i++){
```

```
            if(R[i]>R[i+1]){
                c=R[i+1];
                R[i+1]=R[i];
                R[i]=c;
            };
        };
    };
    if(n%2==0){
        return (R[n/2]+R[(n/2)-1])/2;
    }else{
        return  R[(n-1)/2];
    };
};
```

Problema 61. *Ceros al final.*

Hacer un algoritmo que deje al final de un arreglo de números todos los ceros que aparezcan en dicho arreglo.

probarreblos.h : Linea 21.

probarreblos.cpp : Lineas 190-205.

Modelo Matemático:

$$CerosArr : \mathbb{Z}^* \times \mathbb{N} \to \mathbb{Z}^*$$
$$(Z, n) \mapsto \left\{ (\forall_{i=1}^{n} (\forall Z_i = 0))\, 0 \rightsquigarrow Z_n \wedge \forall_{j=i}^{n} Z_{j+1} \rightsquigarrow Z_j \right.$$

Codificación C++:

```
int* CerosArr (int* Z, int n){
    for(int k = 0;k<n;k++){
        for(int i = 0;i<n;i++){
            if(Z[i]==0){
                for(int j = i;j<n;j++){
                    if(j==n-1){
                        Z[j]=0;
                    }else{
                        Z[j]= Z[j+1];
                    };
                };
            };
        };
    };
```

```
        return Z;
};
```

Problema 62. *De binario a decimal.*

Suponga que un arreglo de enteros esta lleno de unos y ceros y que el arreglo representa un número binario al revés. Hacer un algoritmo que calcule los números en decimal que representa dicho arreglo de unos y ceros.

Modelo Matemático:

$$BinarioDecimalArr : \mathbb{Z}^* \times \mathbb{N} \to \mathbb{Z}$$
$$(Z, n) \mapsto \left\{ \sum_{i=1}^{n} Z_i * 2^i \right.$$

Codificación C++:

```
int BinarioDecimalArr(int* Z, int n){
    int x = 0;
    for(int i = 0; i<n; i++){
        x = x+Z[i]*Potencia(2,i);
    };
    return x;
};
```

Problema 63. *De decimal a binario.*

Hacer un algoritmo que dado un número entero no negativo, cree un arreglo de unos y ceros que representa el número en binario al revés.

Modelo Matemático:

$$DecimalBinarioArr : \mathbb{Z} \to \mathbb{Z}^*$$
$$(x) \mapsto \left\{ \cup a : \forall_n^{n=1} \left(\tfrac{n}{2} + a = x; \tfrac{n}{2} = n \right), n \in \mathbb{N} \right.$$

Codificación C++:

```
int* DecimalBinarioArr(int x){
    int a = x;
    int n = 0;
    while(a>1){
        a/=2;
        n++;
    };
    if(a==1){
```

```
        n++;
    }
    int* numero = new int[n];
    a=x;
    for(int i = 0; i<n;i++){
        numero[i]=a%2;
        a/=2;
    };
    return numero;
};
```

Problema 64. *Máximo común divisor.*

Hacer un algoritmo que calcule el máximo común divisor para un arreglo de enteros positivos.

Modelo Matemático:

$$mcd : \mathbb{N}^* \times \mathbb{N} \to \mathbb{Z}$$

$$(Z, n) \mapsto \begin{cases} x : \forall_{i=0}^{n} \frac{Z_i}{x} \in \mathbb{N} \\ x \text{ es el mayor número natural que cumple la condición.} \end{cases}$$

Codificación C++:

```
int mcd(int* Z, int n){
    int c = MaximoArrEnteros(Z,n);
    int p = 0;
    while (c>0){
        for(int i =0;i<n;i++){
            if(Z[i]%c==0){
                p++;
            };
        };
        if(p==n){
            return c;
        }else{
            c--;
            p=0;
        };
    };
};
```

Problema 65. *Minimo común múltiplo.*

Hacer un algoritmo que calcule el mínimo común multiplo para un arreglo de enteros positivos.

Modelo Matemático:

$$mcm : \mathbb{N}^* \times \mathbb{N} \to \mathbb{Z}$$
$$(Z, n) \mapsto \begin{cases} x : \forall_{i=0}^{n} \frac{x}{Z_i} \in \mathbb{N} \\ x \text{ es el menor número natural que cumple la condición.} \end{cases}$$

Codificación C++:

```
int mcm(int* Z, int n){
    int c = MinimoArrEnteros(Z,n);
    int p = 0;
    while (true){
        for(int i =0;i<n;i++){
            if(c%Z[i]==0){
                p++;
            };
        };
        if(p==n){
            return c;
        }else{
            c++;
            p=0;
        };
    };
};
```

6. Conjuntos como arreglos

Problema 66. *Solicitar al usuario dos conjuntos de números enteros*
Solicitar al usuario el tamaño de dos conjuntos con sus respectivos elementos en número enteros. Asimismo, cada uno de estos conjuntos no contendrán elementos repetidos.

Modelo Matemático:

$$repetido : \mathbb{Z}^* \times \mathbb{Z} \to \mathbb{B}$$
$$(A^*, c) \mapsto \begin{cases} Verdadero, & \text{si } \exists_{i=0}^{c-1}(A_i = A_{i+1}) \\ Falso, & \text{en otro caso.} \end{cases}$$

$$ordenarE : \mathbb{Z}^* \times \mathbb{N} \to \varnothing$$
$$(A^*, n) \mapsto A^* = A_0, A_1, ..., A_{n-1};\ \forall_{i=0}^{n-1}(A_i \leq A_{i+1})$$

$crearC1 : \mathbb{Z} \to \mathbb{Z}^*$

$$(n) \mapsto A^*,\ \forall_{i=0}^{n-1}(repetido(A, c) = Falso\ \wedge\ A_i \leq A_{i+1})$$

$crearC2 : \mathbb{Z} \to \mathbb{Z}^*$

$$(n) \mapsto B^*,\ \forall_{i=0}^{n-1}(repetido(B, c) = Falso\ \wedge\ B_i \leq B_{i+1})$$

Codificación C++:

```
bool repetido(int* A, int c){
    for(int i = 0; i < c; i++){
        for(int j = 0; j < c; j++){
            if(A[i] == A[j] && i != j){
                return true;
            };
        };
    };
    return false;
};

void ordenarE(int* A,int n){
  for(int i = 0; i < n; i++){
    for(int j = i + 1; j < n; j++){
        if(A[j] < A[i]){
            int a = A[i];
            A[i] = A[j];
            A[j] = a;
        };
    };
  };
};

int* crearC1(int n){
    int c = 1;
    int* A = new int[n];
    for(int i = 0; i < n; i++, c++){
        do{
            cout << "Ingrese un numero para el primer conjunto: ";
            cin >> A[i];
        }while(repetido(A,c));
    };
    ordenarE(A,n);
```

```
    return A;
};

int* crearC2(int n){
    int c = 1;
    int* B = new int[n];
    for(int i = 0; i < n; i++, c++){
        do{
            cout << "Ingrese un numero para el segundo conjunto: ";
            cin >> B[i];
        }while(repetido(B,c));
    };
    ordenarE(B,n);
    return B;
};
```

Problema 67. *Unión de conjuntos*
Recibe los dos conjuntos creados por el usuario y los une.

Modelo Matemático:

$$pegarC : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}^*$$
$$(A^*, B^*, n, m) \mapsto C^*, \text{ tal que } C^* = A^* + B^*$$
$$\wedge \ \forall_{i=0}^{n+m-1}(C_i \leq C_{i+1})$$

$$nuevolimiteU : \mathbb{Z}^* \times \mathbb{Z} \to \mathbb{Z}$$
$$(A^*, n) \mapsto c, \ \forall_{i=0}^{n-1}(A_i \neq A_{i+1})$$

$$unionA : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}^*$$
$$(A^*, B^*, n, m) \mapsto D^*, \ \forall_{i=0}^{d-1}((D_i \in A^* \vee D_i \in B^*) \ \wedge \ D_i < D_{i+1})$$

Codificación C++:

```
int* pegarC(int* A, int* B, int n, int m){
    int* C = new int[n + m];
    for(int i = 0; i < n; i++){
        C[i] = A[i];
    };
    for(int i = n, j = 0; i < m || j < m; i++, j++){
        C[i] = B[j];
    };
```

```
    ordenarE(C,n + m);
    return C;
};
int nuevolimiteU(int* A, int n){
    int c = n;
    for(int i = 0; i < n; i++){
      if(A[i] == A[i + 1]){
        c--;
      };
    };
    return c;
};
int* unionA(int* A, int* B, int n, int m){
    int* C = pegarC(A,B,n,m);
    int d = nuevolimiteU(C,n + m);
    int* D = new int[d];
    for(int i = 0, j = 0; i < n + m || j < d; i++){
        if(C[i] != C[i + 1]){
            D[j] = C[i];
            j++;
        };
    };
    delete[] C;
    return D;
};
```

Problema 68. *Intersección de conjuntos*
Recibe los dos conjuntos creados por el usuario y los intersecta.

Modelo Matemático:

$$nuevolimiteI : \mathbb{Z}^* \times \mathbb{Z} \to \mathbb{Z}$$
$$(A^*, n) \mapsto c,\ \forall_{i=0}^{n-1}(A_i = A_{i+1})$$

$$interseccionA : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}^*$$
$$(A^*, B^*, n, m) \mapsto D^*,\ \forall_{i=0}^{d-1}(D_i \in A^* \wedge D_i \in B^*)$$

Codificación C++:

```
int nuevolimiteI(int* A, int n){
    int c = 0;
    for(int i = 0; i < n; i++){
```

```
        if(A[i] == A[i + 1]){
            c++;
        };
    };
    return c;
};

int* interseccionA(int* A, int* B, int n, int m){
    int* C = pegarC(A,B,n,m);
    int d = nuevolimiteI(C,n + m);
    int* D = new int[d];
    for(int i = 0, j = 0; i < n + m || j < d; i++){
        if(C[i] == C[i + 1]){
            D[j] = C[i];
            j++;
        };
    };
    delete[] C;
    return D;
};
```

Problema 69. *Diferencia de conjuntos*
Recibe los dos conjuntos creados por el usuario y calcula la diferencia del primer conjunto con respecto al segundo conjunto.

Modelo Matemático:

$$nuevolimiteD : \mathbb{Z}^* \times \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$$
$$(A^*, n, m) \mapsto c,\ \forall_{i=0}^{n-1}(A_i \neq A_{i+1})$$

$$noperteneceAB : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{Z} \times \mathbb{Z} \to \mathbb{B}^*$$
$$(A^*, B^*, n, m) \mapsto C^*,\ \forall_{i=0}^{n-1}(C_i = Verdadero\ \vee\ C_i = Falso)$$

$$diferenciaA : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}^*$$
$$(A^*, B^*, n, m) \mapsto D^*,\ \forall_{i=0}^{n-1}(noperteneceAB(A, B, n, m)_i = Verdadero)$$

Codificación C++:

```
int nuevolimiteD(int* A, int n, int m){
    int c = m;
    for(int i = 0; i < n; i++){
```

```
        if(A[i] == A[i+1]){
          c--;
        };
    };
    return c;
};
bool* noperteneceAB(int* A, int* B, int n, int m){
    bool* C = new bool[n];
    for(int i = 0; i < n; i++){
        C[i] = true;
    };
    for(int i = 0; i < n; i++){
        for(int j = 0; j < m; j++){
            if(A[i] == B[j]){
                C[i] = false;
            };
        };
    };
    return C;
};
int* diferenciaA(int* A, int* B, int n, int m){
    int* C = pegarC(A,B,n,m);
    int d = nuevolimiteD(C,n+m,n);
    int* D = new int[d];
    for(int i = 0, j = 0; i < n || j < d; i++){
        if(noperteneceAB(A,B,n,m)[i]){
            D[j] = A[i];
            j++;
        };
    };
    delete[] C;
    return D;
};
```

Problema 70. *Diferencia simétrica de conjuntos*
Recibe los dos conjuntos creados por el usuario y calcula la diferencia simétrica de los mismos.

Modelo Matemático:

$$nuevolimiteDS : \mathbb{Z}^* \times \mathbb{Z} \to \mathbb{Z}$$
$$(C^*, n) \mapsto c,\ \forall_{i=0}^{n-1}(C_i \neq C_{i+1})$$

$$simetricaA : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}^*$$
$$(A^*, B^*, n, m) \mapsto D^*, \text{ tal que } D^* = A^* \cup B^* \setminus A^* \cap B^*$$

Codificación C++:

```
int nuevolimiteDS(int* C, int n){
    int c = n;
    for(int i = 0; i < n; i++){
        if(C[i] == C[i + 1]){
            c -= 2;
        };
    };
    return c;
};

int* simetricaA(int* A, int* B, int n, int m){
    int* C = pegarC(A,B,n,m);
    int d = nuevolimiteDS(C,n + m);
    int* D = new int[d];
    for(int i = 0, j = 0; i < n + m || j < d; i++){
        if(C[i] == C[i + 1]){
            i++;
        }else{
            D[j] = C[i];
            j++;
        };
    };
    delete[] C;
    return D;
};
```

Problema 71. *Determinar si un elemento pertenece a algún conjunto*
Recibe los dos conjuntos y un elemento adicional suministrados por el usuario y determina si dicho elemento pertenece a alguno de los dos conjuntos.

Modelo Matemático:

$$perteneceA : \mathbb{Z}^* \times \mathbb{Z} \times \mathbb{Z} \to \mathbb{B}$$
$$(A^*, n, x) \mapsto \begin{cases} Verdadero, & \text{si } \exists_{i=0}^{n-1}(A_i = x) \\ Falso, & \text{en otro caso.} \end{cases}$$

$$perteneceB : \mathbb{Z}^* \times \mathbb{Z} \times \mathbb{Z} \to \mathbb{B}$$

$$(B^*, m, x) \mapsto \begin{cases} Verdadero, & \text{si } \exists_{i=0}^{m-1}(B_i = x) \\ Falso, & \text{en otro caso.} \end{cases}$$

Codificación C++:

```
bool perteneceA(int* A, int n, int x){
    ordenarE(A,n);
    for(int i = 0; i < n; i++){
        if(A[i] == x){
            return true;
        };
    };
    return false;
};

bool perteneceB(int* B, int m, int x){
    ordenarE(B,m);
    for(int i = 0; i < m; i++){
        if(B[i] == x){
            return true;
        };
    };
    return false;
};
```

Problema 72. *Determinar si el primer conjunto está contenido en el segundo conjunto*
Recibe los dos conjuntos creados por el usuario y determina si el primer conjunto está contenido en el segundo conjunto.

Modelo Matemático:

$$contenidoA : \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{Z} \times \mathbb{Z} \to \mathbb{B}$$

$$(A^*, B^*, n, m) \mapsto \begin{cases} Verdadero, & \text{si } \mid \forall_{i=0}^{c-1}(C_i = C_{i+1}) \mid = \mid A^* \mid \\ Falso, & \text{en otro caso.} \end{cases}$$

Codificación C++:

```
bool contenidoA(int* A, int* B, int n, int m){
    int* C = pegarC(A,B,n,m);
    int c = n + m;
```

```
    int d = 0;
    for(int i = 0; i < c; i++){
        if(C[i] == C[i + 1]){
            d++;
        };
    };
    return d == n;
};
```

7. Polinomios como arreglos

Problema 73. *Evaluar un punto en dos polinomios*
Lee un real e imprime la evaluación de los dos polinomios en dicho dato.

Modelo Matemático:

$$f : \mathbb{R}^* \times \mathbb{N} \times \mathbb{R} \to \mathbb{R}$$
$$(A^*, n, R) \mapsto v \;=\; A_n b^n + A_{n-1} b^{n-1} + ... + A_1 b^1 + A_0 b^0$$

Codificación C++:

```
double Xevaluado(double* A,int a,double R){
    double v=0;
    for(int j=a;j>=0;j--){
        v+=A[j]*potenciaot(R,j);
    };
    return v;
};
```

Problema 74. *Sumar dos polinomios*
Calcula el polinomio suma y lo imprime.

Modelo Matemático:

$$f : \mathbb{R}^* \times \mathbb{N} \times \mathbb{R}^* \times \mathbb{N} \to \mathbb{R}$$
$$(A^*, n, B^*, m) \mapsto \; (A_n + B_n)x^n + (A_{n-1} + B_{n-1})x^{n-1} + ... + (A_1 + B_1)x^1 + (A_0 + B_0)x^0$$

Codificación C++:

```
double* SumarPolinomios(double* A,int a,double* B,int b){
    int x=valorabsoluto(a-b);
```

```
    double* nuevo=new double[mayor(a,b)+1];
    for(int i=menor(a,b)+1;i<=mayor(a,b);i++){
        if(a>b){
            nuevo[i]=A[i];
        }else if(b>a){
            nuevo[i]=B[i];
    };
};
for(int i=0;i<=menor(a,b);i++){
       nuevo[i]=A[i]+B[i];
};
    return nuevo;
};
```

Problema 75. *Restar dos polinomios*
Calcula el polinomio resta y lo imprime.

Modelo Matemático:

$$\begin{aligned} f : \mathbb{R}^* \times \mathbb{R} &\to \mathbb{R} \\ (A^*, b) \mapsto a &= (A_n - B_n)x^n + (A_{n-1} - B_{n-1})x^{n-1} + ... \\ &\quad + (A_1 - B_1)x^1 + (A_0 - B_0)x^0 \end{aligned}$$

Codificación C++:

```
double* RestaPolinomios(double* A,int a,double* B,int b){
    int x=valorabsoluto(a-b);
    double* nuevo=new double[mayor(a,b)+1];
    for(int i=menor(a,b)+1;i<=mayor(a,b);i++){
       if(a>b){
         nuevo[i]=A[i];
   }else if(b>a){
     nuevo[i]=B[i];
   };
};
for(int i=0;i<=menor(a,b);i++){
       nuevo[i]=A[i]-B[i];
};
    return nuevo;
};
```

Problema 76. *Multiplicar dos polinomios*
Calcula el polinomio multiplicación y lo imprime.

Modelo Matemático:

$$\begin{aligned} f : \mathbb{R}^* \times \mathbb{R} &\to \mathbb{R} \\ (A^*, b) \mapsto a &= (A_n - B_n)x^n + (A_{n-1} - B_{n-1})x^{n-1} + ... \\ &\quad + (A_1 - B_1)x^1 + (A_0 - B_0)x^0 \end{aligned}$$

Codificación C++:

```
double* Multipolinomios(double* A,int a,double* B,int b){
    int ext=a+b;
    double* nuevo=new double[ext+1];
    for(int i=0;i<=ext;i++){
        nuevo[i]=0;
    };
    for(int j=0;j<=a;j++){
        for(int k=0;k<=b;k++){
            nuevo[j+k]+=A[j]*B[k];
        };
    };
    return nuevo;
};
```

Problema 77. *Dividir dos polinomios*
Calcula el polinomio división y lo imprime.

Modelo Matemático:

$$\begin{aligned} f : \mathbb{R}^* \times \mathbb{R} &\to \mathbb{R} \\ (A^*, a, B*, b) \mapsto a &= (A_n - B_n)x^n + (A_{n-1} - B_{n-1})x^{n-1} + ... \\ &\quad + (A_1 - B_1)x^1 + (A_0 - B_0)x^0 \end{aligned}$$

Codificación C++:

```
double* dividir_pol(double* A,int a,double* B,int b){
        int n=a-b;
        double* X=new double[n+1];
        for(int i=0;i<=n;i++){
            double* K=creararreglo(a-i+1);
            X[n-i]=A[a-i]/B[b];
            K=multicoexdiv(B,b,a-i,X[n-i]);
            A=SumarPolinomios(A,a-i,K,a-i);
        };
```

```
        return X;
};
double* creararreglo(int a){
    double* x=new double[a];
    return x;
};
double* cerosenarreglo(double* A,int n){
    for(int i=0;i<n;i++){
        A[i]=0;
    };
    return A;
};
double* multicoexdiv(double* B, int b,int a,double n){
    double* pro=new double[a+1];
    pro=cerosenarreglo(pro,a+1);
    for(int i=b;i>=0;i--){
        pro[a]=B[i]*(-n);
        a--;
    };
    return pro;
};
```

Problema 78. *Residuo de dos polinomios*
Calcula el polinomio residuo y lo imprime.

Modelo Matemático:

$$\begin{aligned} f : \mathbb{R}^* \times \mathbb{R} &\to \mathbb{R} \\ (A^*, b) \mapsto a &= (A_n - B_n)x^n + (A_{n-1} - B_{n-1})x^{n-1} + ... \\ &\quad + (A_1 - B_1)x^1 + (A_0 - B_0)x^0 \end{aligned}$$

Codificación C++:

```
double* residuo_pol(double* A,int a,double* B,int b){
    int n=a-b;
    double* X=new double[n+1];
    for(int i=0;i<=n;i++){
        double* K=creararreglo(a-i+1);
        X[n-i]=A[a-i]/B[b];
        K=multicoexdiv(B,b,a-i,X[n-i]);
        A=SumarPolinomios(A,a-i,K,a-i);
    };
    return A;
};
```

8. Problemas de matrices.

Problema 79. *Creación de matrices*
Desarrollar un algoritmo que permita crear una matríz, ya sea de reales o enteros.

Modelo Matemático:

$$CrearMatrizInt : \mathbb{N} \times \mathbb{N} \to \mathbb{Z} * *$$
$$(n, m) \mapsto A_{nm}$$

$$CrearMatrizDouble : \mathbb{N} \times \mathbb{N} \to \mathbb{R} * *$$
$$(n, m) \mapsto A_{nm}$$

Codificación C++:

```
int** crear_matriz_int(int n, int m){
    int** X = new int*[n];
    for(int i = 0; i < n; i++){
        X[i] = new int[m];
    };
return X;
};
double** crear_matriz_double(int n, int m){
    double** X = new double*[n];
    for(int i = 0; i < n; i++){
        X[i] = new double[m];
    };
return X;
};
```

Problema 80. *Sumar dos matrices*
Desarrollar un algoritmo que permita sumar dos matrices de números reales y enteros.

Modelo Matemático:

$$mcd : \mathbb{N}^* \times \mathbb{N} \to \mathbb{Z}$$
$$(A^{**}, B^{**}, n, m) \mapsto (A_{nm} + B_{nm}) + (A_{nm-1} + B_{nm-1}) + ... + A_{nm}$$

Codificación C++:

```
int** crear_matriz_int(int n, int m){
    int** X = new int*[n];
    for(int i = 0; i < n; i++){
        X[i] = new int[m];
    };
return X;
};
double** crear_matriz_double(int n, int m){
    double** X = new double*[n];
    for(int i = 0; i < n; i++){
        X[i] = new double[m];
    };
return X;
};
int** SumarMatrices(int** A,int** B,int f,int c){
    int** total=crear_matriz_int(f,c);
    for(int i=0;i<f;i++){
        for(int j=0;j<c;j++){
            total[i][j]=A[i][j]+B[i][j];
        };
    };
    return total;
};
double** SumarMatricesR(double** A,double** B,int f,int c){
    double** total=crear_matriz_double(f,c);
    for(int i=0;i<f;i++){
        for(int j=0;j<c;j++){
            total[i][j]=A[i][j]+B[i][j];
        };
    };
    return total;
};
```

Problema 81. *Multiplicar dos matrices*
Desarrollar un algoritmo que permita multiplicar dos matrices de números reales y enteros.

Modelo Matemático:

$$mcd : \mathbb{N}^* \times \mathbb{N} \to \mathbb{Z}$$
$$(A^{**}, B^{**}, n, m) \mapsto (A_{nm} + B_{nm}) + (A_{nm-1} + B_{nm-1}) + ... + A$$

Codificación C++:

```
int** MultiMatricesE(int** A,int** B,int f,int c){
    int** total=crear_matriz_int(f,c);
    for(int i=0;i<f;i++){
        for(int j=0;j<c;j++){
            total[i][j]=A[i][j]*B[i][j];
        };
    };
    return total;
};
double** MultiMatricesR(double** A,double** B,int f,int c){
    double** total=crear_matriz_double(f,c);
    for(int i=0;i<f;i++){
        for(int j=0;j<c;j++){
            total[i][j]=A[i][j]*B[i][j];
        };
    };
    return total;
};
```

Problema 82. *Sumar columnas de una matríz*
Desarrollar un programa que sume los elementos de una columna dada de una matriz.

Modelo Matemático:

$$mcd : \mathbb{Z}^* \times \mathbb{N} \times \mathbb{N} \to \mathbb{Z}^*$$

$$(X^{**}, n, m) \mapsto \begin{bmatrix} X_{0,1} & X_{0,2} & ... & X_{0,n} \\ + & + & ... & + \\ X_{1,1} & X_{1,2} & ... & X_{1,n} \\ + & + & ... & + \\ X_{2,1} & X_{2,2} & ... & X_{2,n} \\ \| & \| & ... & \| \\ A_0 & A_1 & ... & A_{n-1} \end{bmatrix}$$

Codificación C++:

```
int* SumarColumnas(int**A,int f,int c){
    int* total=new int[c];
    for(int i=0;i<c;i++){
            total[i]=0;
    };
```

```
    for(int i=0;i<c;i++){
        for(int j=0;j<f;j++){
            total[i]+=A[j][i];
        };
    };
    return total;
};
```

Problema 83. *Sumar filas de una matríz*
Desarrollar un programa que sume los elementos de una fila dada de una matriz.

Modelo Matemático:

$$sumarFilas : \mathbb{Z}^* \times \mathbb{N} \times \mathbb{N} \to \mathbb{Z}^*$$

$$(X^{**}, n, m) \mapsto \begin{bmatrix} X_{0,1} + & X_{0,2} + & ... & + X_{0,n} = A_0 \\ X_{1,1} + & X_{1,2} + & ... & + X_{1,n} = A_1 \\ ... & ... & ... & ... \quad ... \\ X_{m,1} + & X_{m,2} + & ... & + X_{m,n} = A_m \end{bmatrix}$$

Codificación C++:

```
int* SumarFilas(int**A,int f,int c){
    int* total = new int[f];
    for(int i = 0; i < f; i++){
            total[i] = 0;
    };
    for(int i = 0; i < f; i++){
        for(int j = 0; j < c; j++){
            total[i] += A[i][j];
        };
    };
    return total;
};
```

Problema 84. *Cuadrado mágico*
Desarrollar un algoritmo que determine si una matriz es mágica. Se dice que una matriz cuadrada es mágica si la suma de cada una de sus filas, de cada una de sus columnas y de cada diagonal es igual.

Modelo Matemático:

$$CuadradoMagico : \mathbb{Z}^{**} \times \mathbb{N} \to \mathbb{B}$$

$$(A^{**}, n) \mapsto \begin{cases} Verdadero, & d1 = d2 \wedge perteneceA(F, n, d1); \\ Falso, & \text{en otro caso .} \end{cases}$$

Codificación C++:

```
bool CuadradoMagico(int**A, int n){
    int* F = new int[n];
    int* C = new int[n];
    int d1 = 0;
    int d2 = 0;
    for(int i = 0; i < n; i++){
        F[i] = 0;
    };
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n; j++){
            F[i] += A[i][j];
        };
    };
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n; j++){
            if(F[i] != F[j]){
                return false;
            };
        };
    };
    for(int i = 0; i < n; i++){
        C[i] = 0;
    };
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n; j++){
            C[i] += A[j][i];
        };
    };
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n; j++){
            if(C[i] != C[j]){
                return false;
            };
        };
    };
    for(int i = 0; i < n; i++){
        if(F[i] != C[i]){
            return false;
        };
    };
```

```
    for(int i = 0; i < n; i++){
        d1 += A[i][i];
    };
    for(int i = n - 1, j = 0; i >= 0 && j < n; i--, j++){
        d2 += A[i][j];
    };
    return d1 == d2 && perteneceA(F,n,d1);
};
```

Problema 85. *Reemplazar por unos y ceros la matriz*
Desarrollar un algoritmo que dado un entero, reemplace en una matriz todos los números mayores a un número dado por un uno y todos los menores o iguales por un cero.

Modelo Matemático:

$$ReemplazarM : \mathbb{Z}^{**} \times \mathbb{N} \times \mathbb{N} \times \mathbb{Z} \to \mathbb{Z}^{**}$$
$$(A^{**}, n, m, x) \mapsto A^{**}, \forall_{i=0}^{n-1} \forall_{j=0}^{m-1} (A_{i,j} > x \mapsto 1 \wedge A_{i,j} \leq x \mapsto 0)$$

Codificación C++:

```
int** ReemplazarM(int** A, int filas, int columnas, int x){
    for(int i = 0; i < filas; i++){
        for(int j = 0; j < columnas; j++){
            if(A[i][j] > x){
                A[i][j] = 1;
            }else{
                A[i][j] = 0;
            };
        };
    };
    return A;
};
```

Problema 86. *Derterminante de una matriz*

Desarrollar un programa que calcule el determinante de una matríz cuadrada.

Modelo Matemático:

$$DeterminanteMatriz : \mathbb{Z}^{**} \times \mathbb{N} \to \mathbb{Z}$$
$$(X, n) \mapsto \begin{cases} X_1, \\ X_{1,1}X_{2,2} - X_{1,2}X_{2,1}, \\ \sum_{i=1}^{n} X_{1,i} * -1^{i+1} * DeterminanteMatriz(Adjunta(X, i, n-1)), \end{cases}$$

Codificación C++:

```
int DeterminanteMatriz(int** X,int n){
    if(n==2){
        return X[0][0]*X[1][1]-(X[1][0]*X[0][1]);
    }else{
        int a=0;
        for(int i =0;i<n;i++){
            a+=X[0][i]*Potencia(-1,i)*DeterminanteMatriz(Adjunta(X,i,n-1),n-1);
        };
        return a;
    };
};
```

Problema 87. *Matriz Corrida verticalmente*

Modelo Matemático:

$MatrizCorrida : \mathbb{Z}^{**} \times \mathbb{N} \times \mathbb{N} \to \mathbb{Z}^{**}$

$$(X, f, n) \mapsto \begin{cases} \begin{bmatrix} X_{f,1} & X_{f,2} & X_{f,3} & X_{f,4} & ... & X_{f,n} \\ X_{f+1,1} & X_{f+1,2} & X_{f+1,3} & X_{f+1,4} & ... & X_{f+1,n} \\ X_{f+2,1} & X_{f+2,2} & X_{f+2,3} & X_{f+2,4} & ... & X_{f+2,n} \\ ... & ... & ... & ... & ... & ... \\ X_{f-1,2} & X_{f-1,3} & X_{f-1,c-1} & X_{f-1,c+1} & ... & X_{f-1,n} \end{bmatrix} \end{cases}$$

Codificación C++:

```
int** MatrizCorrida(int** X,int f,int n){
    int** A = new int*[n];
    for(int i =0;i<n;i++){
        A[i]=new int[n];
    };
    if(f==0){
        return X;
    }else{
        for(int i =0;i<n-1;i++){
            for(int j=0;j<n;j++){
                A[i][j]=X[i+1][j];
            };
        };
        for(int j=0;j<n;j++){
```

```
            A[n-1][j]=X[0][j];
        };
        return MatrizCorrida(A,f-1,n);
    };
};
```

Problema 88. *Matriz Traspuesta de una matriz cuadrada*

Modelo Matemático:

$$MatrizTraspuesta : \mathbb{Z}^{**} \times \mathbb{N} \to \mathbb{Z}^{**}$$
$$(X, n) \mapsto \left\{ \forall_{i=0}^{n} \forall_{j=0}^{n} X_{i,j} = X_{j,i} \right.$$

Codificación C++:

```
int** MatrizTraspuesta(int** X, int n){
    int** Y = new int*[n];
    for(int i =0;i<n;i++){
        Y[i]= new int[n];
    };
    for(int i=0;i<n;i++){
        for(int j =0;j<n;j++){
            Y[i][j]=X[j][i];
        };
    };
    return Y;
};
```

Problema 89. *Matriz Inversa*

Modelo Matemático:

$$MatrizInversa : \mathbb{Z}^{**} \times \mathbb{N} \to \mathbb{Z}^{**}$$
$$(X, n) \mapsto \begin{cases} \begin{bmatrix} -X_{2,2} & X_{1,2} \\ X_{2,1} & -X_{1,1} \end{bmatrix}, \\ \forall_{i=0}^{n} \forall_{j=0}^{n} : X_{i,j} = -1^{j} * DeterminanteMatriz(Adjunta(MatrizTraspues \end{cases}$$

Codificación C++:

```
double** MatrizInversa(int** X, int n){
    double det = DeterminanteMatriz(X,n);
    double** A = new double*[n];
    for(int i =0;i<n;i++){
        A[i]=new double[n];
    };
    if(n==2){
        A[0][0]= (X[1][1])/det;
        A[0][1]= -(X[0][1])/det;
        A[1][0]= -(X[1][0])/det;
        A[1][1]= (X[0][0])/det;
        return A;
    }else{
        for(int i=0;i<n;i++){
            for(int j =0;j<n;j++){
                A[i][j]=Potencia(-1,j)*(DeterminanteMatriz(Adjunta(MatrizTraspuesta(X,n
            };
        };
        return A;
    };
};
```

Problema 90. *Adjunta de la posición del componente de una matríz en la primera fila*

Modelo Matemático:

$$Adjunta : \mathbb{Z}^{**} \times \mathbb{N} \times \mathbb{N} \to \mathbb{Z}^{**}$$

$$(X, c, n) \mapsto \begin{cases} \begin{bmatrix} X_{2,2} & X_{2,3} & X_{2,c-1} & X_{2,c+1} & \dots & X_{2,n} \\ X_{3,2} & X_{3,3} & X_{3,c-1} & X_{3,c+1} & \dots & X_{3,n} \\ X_{4,2} & X_{4,3} & X_{4,c-1} & X_{4,c+1} & \dots & X_{4,n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ X_{n,2} & X_{n,3} & X_{n,c-1} & X_{n,c+1} & \dots & X_{n,n} \end{bmatrix} \end{cases}$$

Codificación C++:

```
int** Adjunta(int**X, int c, int n){
    int** A = new int*[n];
    for(int i =0;i<n;i++){
        A[i]=new int[n];
```

```
    };
    int y=0;
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            if(c==j){
                y=1;
                A[i][j]=X[i+1][j+y];
            }else{
                A[i][j]=X[i+1][j+y];
            };
        };
        y=0;
    };
    return A;
};
```

Problema 91. *Adjunta de un componente determinado de una matríz*

Modelo Matemático:

$$Adjunta : \mathbb{Z}^{**} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{Z}^{**}$$

$$(X, f, c, n) \mapsto \begin{cases} Adjunta(X, c, n-1), & n = 1 \\ Adjunta(MatrizCorrida(X, f, n), c, n-1), & \text{En otro caso} \end{cases}$$

Codificación C++:

```
int** Adjunta(int**X,int f, int c, int n){
    if(f==0){
        return Adjunta(X,c,n-1);
    }else{
        return Adjunta(MatrizCorrida(X,f,n),c,n-1);
    };
};
```

Problema 92. *Girar matriz cuadrada 90 grados en el sentido contrario de las manecillas del reloj*

Modelo Matemático:

$$GirarMatrizContManecillas : \mathbb{Z}^{**} \times \mathbb{N} \to \mathbb{Z}^{**}$$

$$(X, n) \mapsto \left\{ \forall_{i=0}^{n} \forall_{j=0}^{n} : X_{i,j} = X_{j,n-i-1} \right.$$

Codificación C++:

```
int** GirarMatrizContManecillas(int** X, int n){
    int** A = new int*[n];
    for(int i =0;i<n;i++){
        A[i]=new int[n];
    };
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
                A[i][j]= X[j][n-1-i];
        };
    };
    return A;
};
```

Problema 93. *Solución al sistema de ecuaciones nxn*

Modelo Matemático:

$$SistemaEcuaciones : \mathbb{R}^{**} \times \mathbb{R}^{*} \times \mathbb{N} \to \mathbb{R}^{*}$$
$$(X, S, n) \mapsto \left\{ T : \forall_{i=0}^{n} : T_i = \frac{MatrizReemplazada(X, S, i, n)}{DeterminanteMatriz(X, n)} \right.$$

Codificación C++:

```
double* SistemaEcuaciones(int**X, int* S, int n){
    double* T = new double[n];
    int det = DeterminanteMatriz(X,n);
    for(int i =0;i<n;i++){
        T[i]=DeterminanteMatriz(MatrizReemplazada(X,S,i,n),n)/det;
    };
    return T;
};
```

Problema 94. *Matriz con una columna reemplazada por un arreglo*

Modelo Matemático:

$$MatrizReemplazada : \mathbb{Z}^{**} \times \mathbb{Z}^{*} \times \mathbb{N} \times \mathbb{N} \to \mathbb{Z}^{*}$$

$$(X, S, i, n) \mapsto \begin{cases} \begin{bmatrix} X_{0,0} & X_{0,i-1} & S_0 & X_{0,i+1} & ... & X_{0,n} \\ X_{1,0} & X_{1,i-1} & S_1 & X_{1,i+1} & ... & X_{1,n} \\ X_{2,0} & X_{2,i-1} & S_2 & X_{2,i+1} & ... & X_{2,n} \\ ... & ... & ... & ... & ... & ... \\ X_{n,0} & X_{n,i-1} & S_3 & X_{n,i+1} & ... & X_{n,n} \end{bmatrix} \end{cases}$$

Codificación C++:

```
int** MatrizReemplazada(int** X, int* S, int k, int n){
    int** A = CrearMatrizInt(n,n);
    for(int i =0;i<n;i++){
        for(int j =0;j<k;j++){
            A[i][j]=X[i][j];
        };
    };
    for(int i=0;i<n;i++){
        A[i][k]=S[i];
    };
    for(int i =0;i<n;i++){
        for(int j =k+1;j<n;j++){
            A[i][j]=X[i][j];
        };
    };
    return A;
};
```

Problema 95. *Matriz en espiral*

Modelo Matemático:

$$Espiral : \mathbb{Z}^{*} \times \mathbb{N} \to \mathbb{Z}^{**}$$

$$(X, n^2) \mapsto \left\{ \forall_{i=0}^{2n-1} \left(\forall_{j=0}^{n-p} : X_{o,j+m} = X_{cont}; cont = cont + 1 \right); p = p + \tfrac{1}{2}; m = m + \tfrac{1}{4}; o + 1 = o \right.$$

Codificación C++:

```
int** Espiral(int* X,int n){
    int a = Raiz2(n);
```

```
    int** A = CrearMatrizInt(a,a);
    for(int i=0;i<a;i++){
        for(int j=0;j<a;j++){
            A[i][j]=0;
        };
    };
    int m = 0;
    int p = 0;
    int o = 0;
    int cont = 0;
    for(int i=0;i<2*a-1;i++){
        for(int j = 0;j<a-p;j++){
            A[o][j+m]=X[cont];
            cont++;
        };
        A=GirarMatrizContManecillas(A,a);
        if(i%2==0){
            p++;
        };
        if(i%4==0){
            m++;
        };
        if((i+1)%4==0){
            o++;
        };
    };
    int b=(2*a-1)%4;
    for(int i=0; i<4-b;i++){
        A=GirarMatrizContManecillas(A,a);
    };
    return A;
};
```