# Programación Orientada a Objetos
# Entrega obligatoria

1. **Realizar una clase que permita representar una fracción.**
    - Definir los datos miembros de la clase.
    - Definir si fuera necesario funciones de carga y muestra de los datos miembro.
    - Definir uno o más constructores.
    - Definir los siguientes métodos:
        - Sumar: calcula la suma de 2 racionales generando otro como resultado.
        - Restar: calcula la resta de 2 racionales generando otro como resultado.
        - Incrementar: incrementa en 1 un racional
        - Decrementar: decrementa en 1 un racional
        - Simplificar: calcula la fracción simplificada

    Realizar un programa principal que haga uso de la clase

**I. Se provee la siguiente solución:**

**a) Prototipo de la clase, archivo racional.h**

```
class racional {
 private:
 int num;
 int den;
 int error;
 int MCD();
 public:
  racional();
  racional(int, int);
  void setNum(int);
  int getNum();
  void setDen(int);
  int getDen();
  int getError();
  void incrementar();
  void decrementar();
  racional sumar(racional);
  racional restar(racional);
  void simplificar();
 };
```

**b) Implementación de la clase, archivo racional.cpp**

```
#include <math.h>
#include "racional.h"

 racional::racional(){
  num=0;
  den=1;
  error=0;
 }
 racional::racional(int n, int d){
  num=n;
  den=d;
  if (den!=0) error=0;
  else error=1;
 }
 void racional::setNum(int n){
  num=n;
 }
 int racional::getNum(){
  return num;
 }
 void racional::setDen(int d){
   den=d;
   if (den!=0) error=0;
   else error=1;
 }
 int racional::getDen(){
  return den;
 }
 int racional::getError(){
  return error;
 }
 void racional::incrementar(){
  num+=den;
  simplificar();
  }
 void racional::decrementar(){
    num-=den;
    simplificar();
  }
 racional racional::sumar(racional x){
   racional z(num*x.den+den*x.num, den*x.den);
   z.simplificar();
   return z;
 }
 racional racional::restar(racional x){
   racional z(num*x.den-den*x.num, den*x.den);
```

```
  z.simplificar();
  return z;
 }
 void racional::simplificar(){
 int d=MCD();
 num/=d;
 den/=d;
 }

 int racional::MCD(){
  int a,b,aux,r;
  a=num; b=den;
  if (a<0) a*=-1;
  if (b<0) b*=-1;
  if (b>a) {
     aux=a;
     a=b;
     b=aux;
   }
   r=a%b;
   while (r!=0){
      a=b;
      b=r;
      r=a%b;
    }
    return b;
  }
```

**c) Programa principal, archivo main_racional.cpp**

```
#include "racional.h"
using namespace std;
int main() {
 racional a(1,2), b(4,3);
 cout<<"a = "<<a.getNum()<<"/"<<a.getDen()<<"\n";
 cout<<"b = "<<b.getNum()<<"/"<<b.getDen()<<"\n";
 a.incrementar();
 cout<<"El racional a incrementado en 1 ";
 cout<<"a = "<<a.getNum()<<"/"<<a.getDen()<<"\n";
 b.decrementar();
 cout<<"El racional b decrementado en 1 ";
 cout<<"b = "<<b.getNum()<<"/"<<b.getDen()<<"\n";
 cout<<"La suma de a y b es C ";
 racional c=a.sumar(b);
 cout<<"c = "<<c.getNum()<<"/"<<c.getDen()<<"\n";
 cout<<"La resta de c  y b es d ";
 racional d=c.restar(b);
 cout<<"d = "<<d.getNum()<<"/"<<d.getDen()<<"\n";
}
```

**II. Sobre la solución provista realizar lo siguiente:**

a) Analizar el código provisto y explicar la funcionalidad de cada uno de los métodos provistos.
b) Crear un proyecto para vincular los códigos anteriores.
c) Probar la ejecución y verificar si la solución es correcta

**2. Realizar una clase que permita representar una fecha.**

- Definir los datos miembros de la clase.
- Definir si fuera necesario funciones de carga y muestra de los datos miembro.
- Definir un constructor que inicializa la fecha a una fecha dada.
- Definir un constructor que inicializa la fecha en 01/01/1900.
- Definir los siguientes métodos:
- ayer: decrementa la fecha en 1 día.
- manana: incrementa la fecha en 1 día.
- Sumar: calcula la fecha resultante de sumar a una fecha una cierta cantidad de días.
- Restar: calcula la fecha resultante de restar a una fecha una cierta cantidad de días.

**I. Se provee el prototipo de la clase, parte de la implementación y el programa principal:**

**Fecha.h**

```
class Fecha{
 private:
   int dia;
   int mes;
   int anio;
   int dias(int);
 public:
   Fecha();
   Fecha(int, int, int);
   void setDia(int);
   void setMes(int);
   void setAnio(int);
   int getDia();
   int getMes();
   int getAnio();
   void ayer();
   void manana();
   int valida();
   Fecha sumar(int );
   Fecha restar(int);
};
```

**Fecha.cpp**

```
#include "fecha.h"
Fecha::Fecha(){
 dia=1;
 mes=1;
 anio=1900;
}
Fecha::Fecha(int d, int m, int a){
 dia=d;
 mes=m;
 anio=a;
}
void Fecha::setDia(int d)
{
   dia=d;
}
void Fecha::setMes(int m)
{
   mes=m;
}
void Fecha::setAnio(int a)
{
   anio=a;
}
int Fecha::getDia()
{
   return dia;
}
int Fecha::getMes()
{
   return mes;
}
int Fecha::getAnio()
{
   return anio;
}
int Fecha::valida()
{
   if (dia<1 ||dia>31 ||mes<1 ||mes>12) return 0;
   if (dia>30 && (mes==4||mes==6||mes==9||mes==11)) return 0;
   if (dia>29 && mes==2) return 0;
   if (dia>28 && mes==2 && anio%4!=0) return 0;
   return 1;
}

void Fecha::manana(){
 dia++;
 if (!valida()){
```

```
      dia=1;
      mes++;
      if (!valida()){
         mes=1;
         anio++;
      }
     }
    }
    void Fecha::ayer(){
    ….
    }
    int Fecha::dias(int m){
    ..
    }
    Fecha Fecha::sumar(int d){
    ..
    }
    Fecha Fecha::restar(int d){
    ..
    }
```

**Main_Fecha.cpp**

```
#include <iostream>
#include "fecha.h"
using namespace std;
  int main() {
  int d, m, a, ig=0;
  Fecha f1(1,1, 2010) ;
  Fecha f2(31,12, 2010) ;
  Fecha f3, f4;

  cout<<"\n"<<"fecha uno "<<f1.getDia()<<"/"<<f1.getMes()<<"/"<<f1.getAnio();
  f1.ayer();
  cout<<"\n"<<"fecha uno menos 1 dia es "<<f1.getDia()<<"/"<<f1.getMes()<<"/"<<f1.getAnio();

  cout<<"\n"<<"fecha dos "<<f2.getDia()<<"/"<<f2.getMes()<<"/"<<f2.getAnio();
  f2.manana();
  cout<<"\n"<<"fecha uno mas un dia es "<<f2.getDia()<<"/"<<f2.getMes()<<"/"<<f2.getAnio();

  f3=f1.sumar(10);
  cout<<"\n"<<"fecha uno mas 10 dias es "<<f3.getDia()<<"/"<<f3.getMes()<<"/"<<f3.getAnio();
  f4=f2.restar(25);
  cout<<"\n"<<"fecha dos menos 25 dias es "<<f4.getDia()<<"/"<<f4.getMes()<<"/"<<f4.getAnio();
}
```

**II. A partir de la solución provista realizar lo siguiente:**

a) Completar la implementación de la clase implementando las funciones faltantes.
b) Crear un proyecto para vincular los códigos anteriores.
c) Probar la ejecución y verificar si la solución es correcta