



# Práctica Final.

Rutas Áreas

Rosa M<sup>a</sup> Rodríguez Sánchez.

# Objetivos



- Haber estudiado el Tema 1: Introducción a la eficiencia de los algoritmos
- Haber estudiado el Tema 2: Abstracción de datos. Templates.
- Haber estudiado el Tema 3: T.D.A. Lineales.
- Haber estudiado el Tema 4: STL e Iteradores.
- Haber realizado la Práctica 2: TDA Imagen. En el caso que el estudiante no haya trabajado esta práctica es recomendable entenderla.

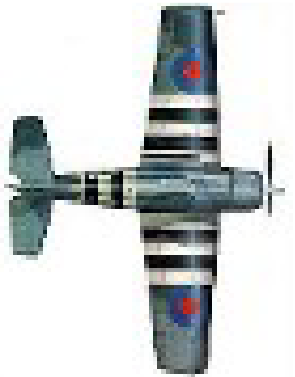


# Tareas a realizar

1. Rotar: Programa que permita rotar una imagen de color y con transparencia.
2. Pegar: Programa que permita pegar una imagen en otra.
3. Pintar Ruta: Programa que dada una ruta área nos pinte en un mapa del mundo tal ruta.

# Tarea 1.- Rotar una imagen

Input



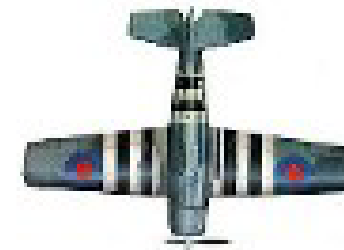
$$\begin{pmatrix} \cos(\alpha) & \sin(\alpha) \\ \sin(-\alpha) & \cos(\alpha) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix}$$

$$I_{out}(x', y') = I_{input}(x, y)$$

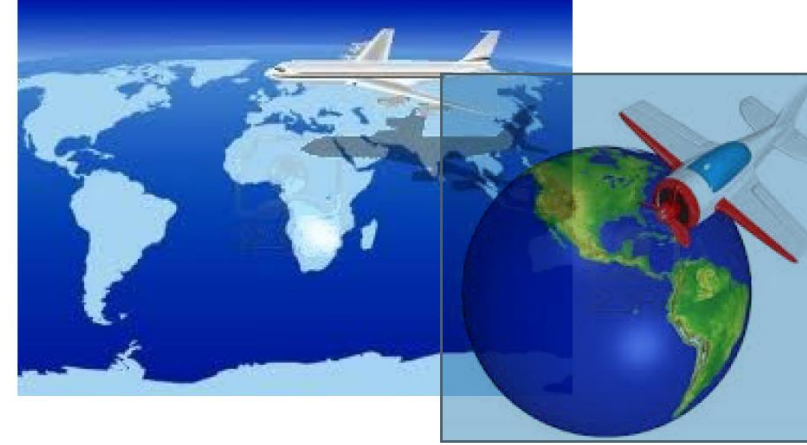
Output



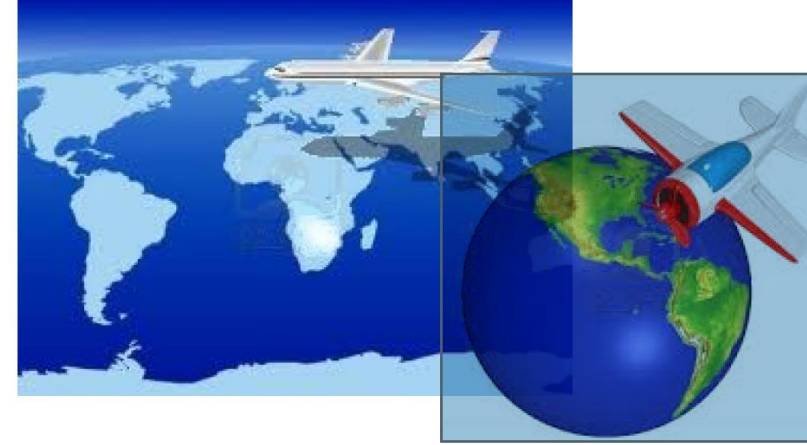
$\alpha = 45^\circ$



$\alpha = 90^\circ$



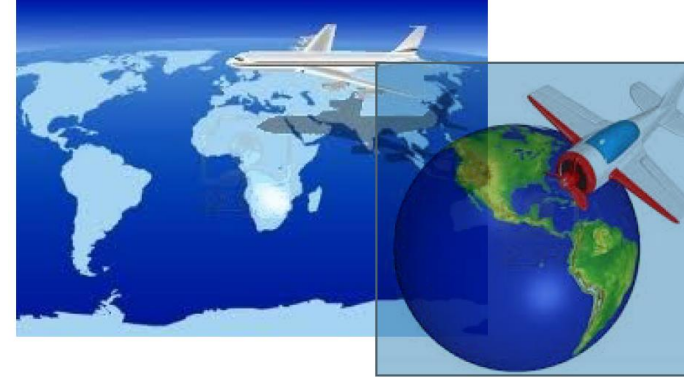
# Tarea 1.- Rotar una imagen



- ¿Qué necesitamos?
  1. Módulo Imagen
  2. Fichero probarotación.cpp donde debe estar la función Rotar.

# Módulo Imagen.

Son imágenes de color que almacenan transparencia.

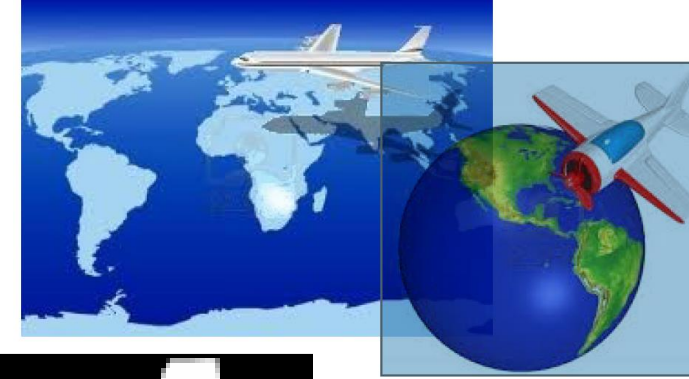
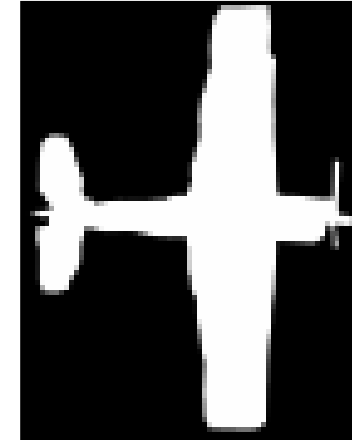


```
7  enum Tipo_Pegado {OPACO, BLENDING};
8  using namespace std;
9  struct Pixel{
10     unsigned char r,g,b;
11     unsigned char transp; //0 no 255 si
12 };
13 class Imagen{
14     private:
15         Pixel ** data;
16         int nf,nc;
17         void Borrar();
18         void Copiar(const Imagen &I);
19     public:
20         Imagen();
21         Imagen(int f,int c);
22         Imagen(const Imagen & I);
```

```
23     Imagen & operator=(const Imagen & I);
24     ~Imagen();
25     //set y get
26     Pixel & operator () (int i,int j);
27     const Pixel & operator () (int i,int j) const;
28     void EscribirImagen(const char * nombre);
29     void LeerImagen (const char *nombre,const string &nombremascara="");
30     void LimpiarTransp();
31     int num_filas() const{return nf;}
32     int num_cols() const{return nc;}
33     void PutImagen(int posi,int posj, const Imagen &I,Tipo_Pegado tippegado=OPACO);
34     Imagen ExtraeImagen(int posi,int posj,int dimi,int dimj);
35 };
36
```

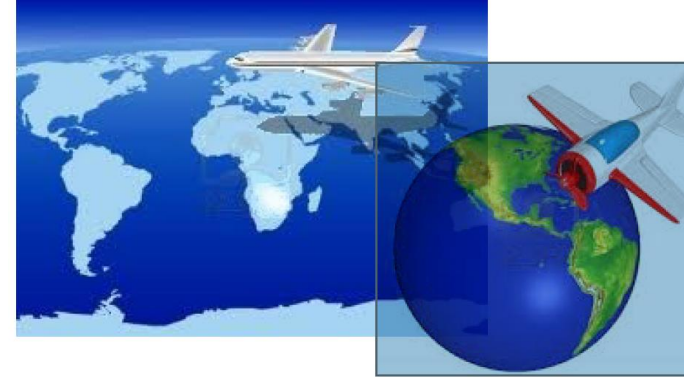
# Módulo Imagen. Transparencia

```
Imagen::Imagen(int f,int c){  
    nf = f;  
    nc = c;  
    data = new Pixel*[nf];  
    for (int i=0;i<nf;i++){  
        data[i]=new Pixel[nc];  
        for (int j=0;j<nc;j++){  
            data[i][j].r=255;  
            data[i][j].g=255;  
            data[i][j].b=255;  
            data[i][j].transp=255;  
        }  
    }  
}
```

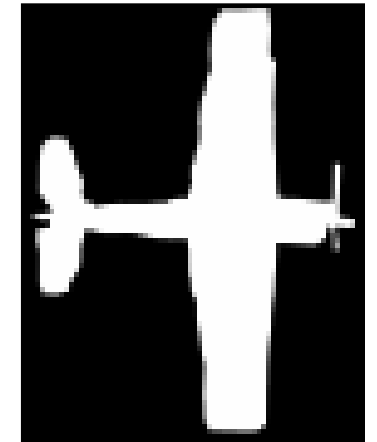
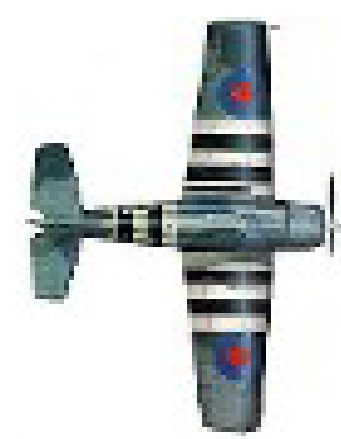




# Módulo Imagen. Transparencia



```
105 void Imagen::LeerImagen(const char * nombre, const string &nombremascar  
106     int f, c;  
107     unsigned char * aux, *aux_mask ;  
108     LeerTipoImagen(nombre, f, c);  
109     aux = new unsigned char [f*c*3];  
110     LeerImagenPPM (nombre, f, c, aux);  
111     if (nombremascara!="") {  
112         int fm, cm;  
113         LeerTipoImagen(nombremascara.c_str(), fm, cm);  
114         aux_mask = new unsigned char [fm*cm];  
115         LeerImagenPGM(nombremascara.c_str(), fm, cm, aux_mask);  
116     }  
117     else {  
118         aux_mask=0;  
119     }  
120     Imagen I(f, c);  
121     int total = f*c*3;  
122     for (int i=0; i<total; i+=3) {  
123         int posi = i / (c*3);  
124         int posj = (i%(c*3))/3;  
125         I.data[posi][posj].r=aux[i];  
126         I.data[posi][posj].g=aux[i+1];  
127         I.data[posi][posj].b=aux[i+2];  
128         if (aux_mask!=0)  
129             I.data[posi][posj].transp=aux_mask[i/3];  
130         else  
131             I.data[posi][posj].transp=255;  
132     }  
133     *this = I;  
134     if (aux_mask!=0) delete [] aux_mask;  
135     delete [] aux;  
136 }
```



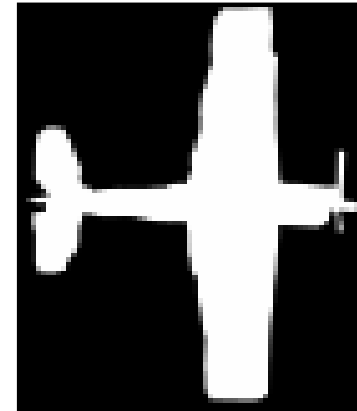
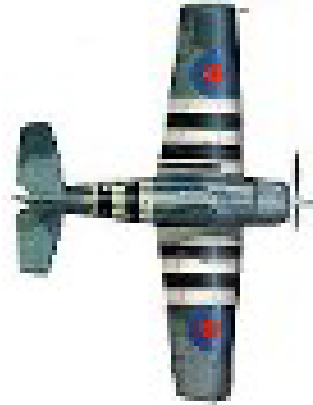
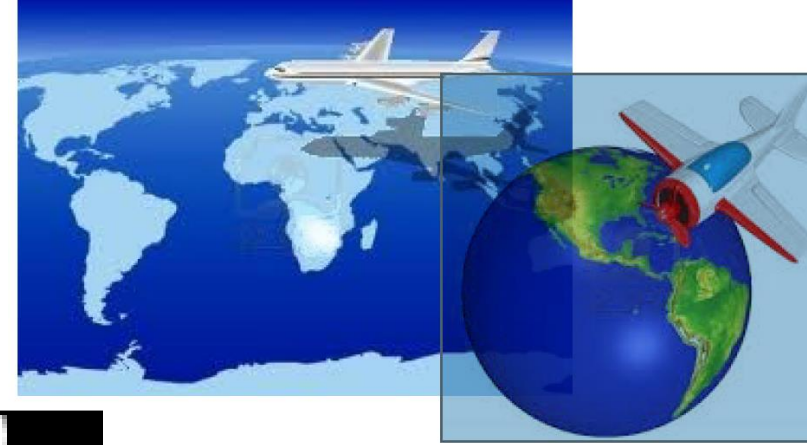


# Tarea 1.- Rotar una imagen



```
2  #include <iostream>
3  #include <cmath>
4  #include "imagen.h"
5  using namespace std;
6  Imagen Rota(const Imagen & Io, double angulo) {
74
75
76  int main(int argc, char * argv[]) {
77      if (argc!=4) {
78          cout<<"Los parametros son :"<<endl;
79          cout<<"1.-La imagen de entrada"<<endl;
80          cout<<"2.-El angulo de rotación"<<endl;
81          cout<<"3.-El nombre de la imagen de salida"<<endl;
82          return 0;
83      }
84      Imagen I;
85      I.LeerImagen(argv[1]);
86      double angulo=atof(argv[2]);
87      angulo = angulo*(M_PI)/180;
88      Imagen Iout=Rota(I, angulo);
89      Iout.EscribirImagen(argv[3]);
90
91  }
```

## Tarea 2.- Pegar una imagen



Pegado Opaco

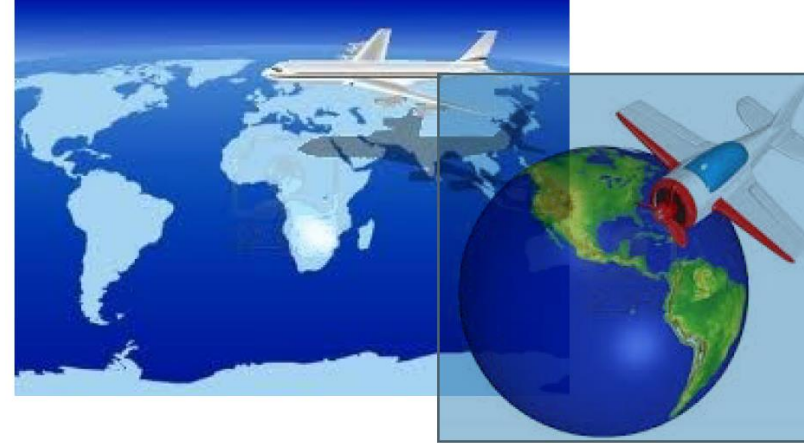


Pegado Blending



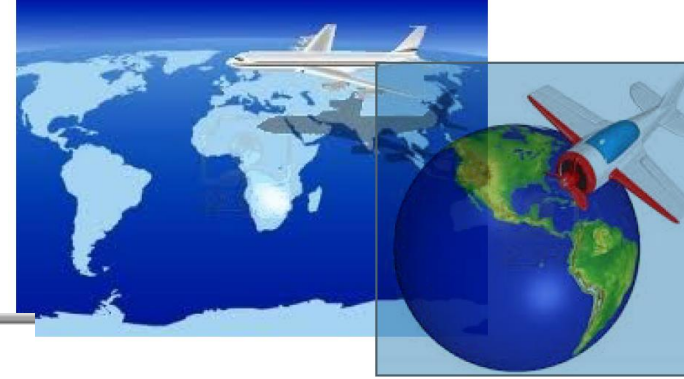
# Tarea 2.- Pegar una imagen

- ¿Qué necesitamos?
  1. Módulo Imagen
  2. Fichero pruepegado.cpp



# Módulo Imagen.

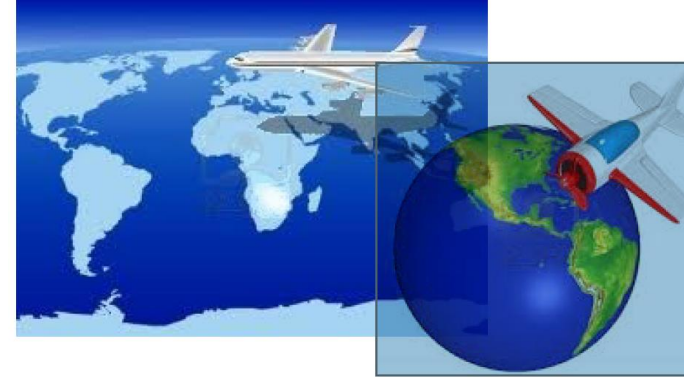
Son imágenes de color que almacenan transparencia.



```
1  enum Tipo_Pegado {OPACO, BLENDING};
2  using namespace std;
3  struct Pixel{
4      unsigned char r,g,b;
5      unsigned char transp; //0 no 255 si
6  };
7  class Imagen{
8  private:
9      Pixel ** data;
10     int nf,nc;
11     ...
12 public:
13     ....
14     void PutImagen(int posi,int posj, const Imagen &I,
15                   Tipo_Pegado tippegado=OPACO);
16
17     ...
18 };
19
```

# Módulo Imagen.

Son imágenes de color que almacenan transparencia.



```
147 void Imagen::PutImagen(int posi,int posj, const Imagen &I,Tipo_Pegado tippegado){
148     //assert(nf>=posi+I.nf && nc>=posj+I.nc);
149
150     for (int i=0;i<I.nf;i++)
151         for (int j=0;j<I.nc;j++)
152             if (i+posi>=0 && i+posi<nf && j+posj>=0 && j+posj<nc){
153                 if (I.data[i][j].transp!=0){
154                     if (tippegado==OPACO)
155                         data[i+posi][j+posj]=I.data[i][j];
156                     else{
157                         data[i+posi][j+posj].r=(data[i+posi][j+posj].r+I.data[i][j].r)/2;
158                         data[i+posi][j+posj].g=(data[i+posi][j+posj].g+I.data[i][j].g)/2;
159                         data[i+posi][j+posj].b=(data[i+posi][j+posj].b+I.data[i][j].b)/2;
160                     }
161                 }
162             }
163 }
```

## Tarea 2.- Pegar una imagen

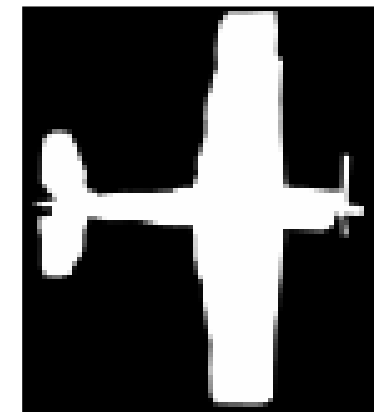
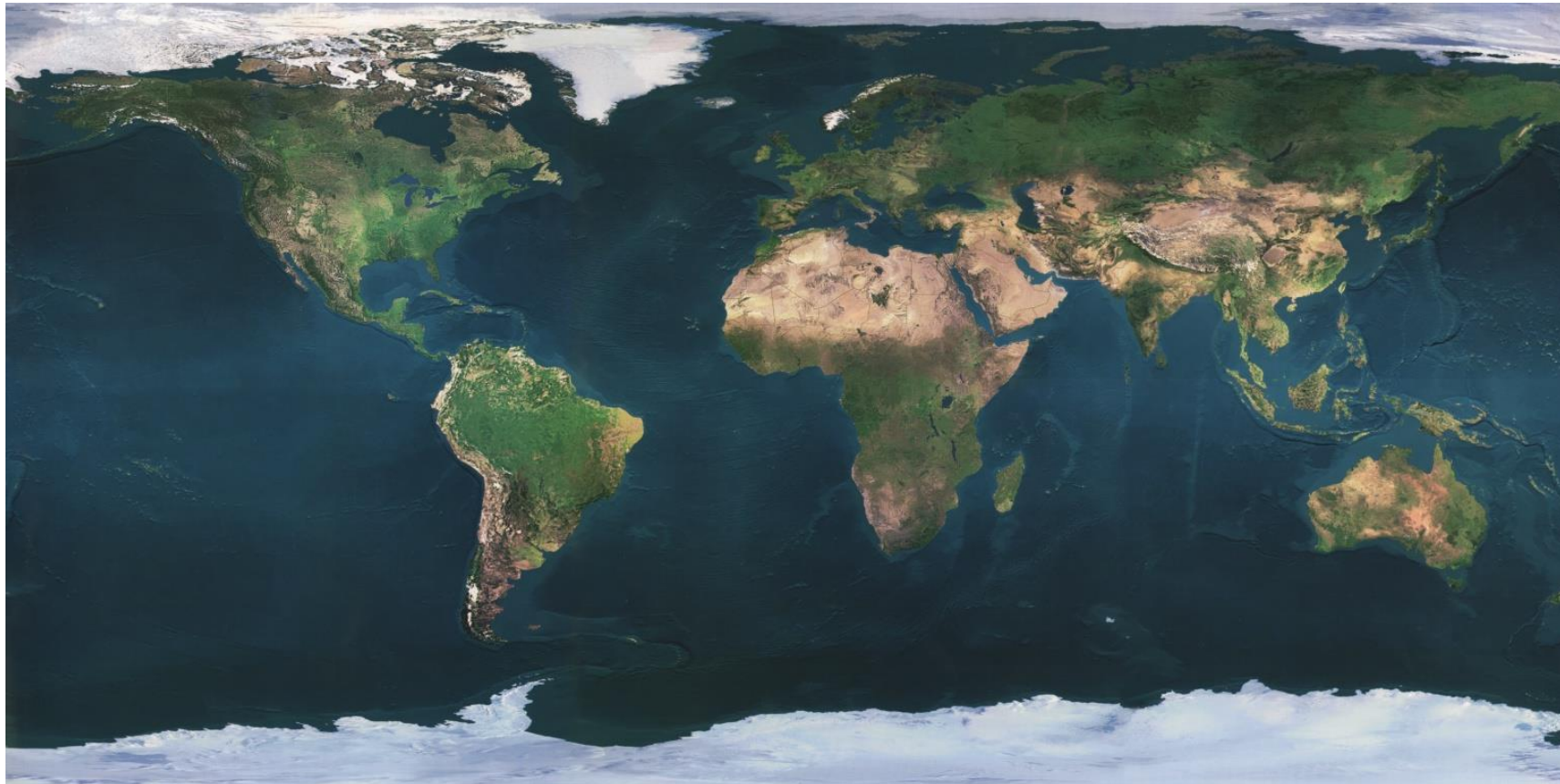
```
5  int main(int argc, char * argv[]){
6      if (argc!=8){
7          cout<<"Los parametros son :"<<endl;
8          cout<<"1.-La imagen de fondo"<<endl;
9          cout<<"2.-La imagen a pegar"<<endl;
10         cout<<"3.-La máscara de la imagen a pegar"<<endl;
11         cout<<"4.-El nombre de la imagen de salida"<<endl;
12         cout<<"5.-La fila donde pegar"<<endl;
13         cout<<"6.-La columna donde pegar"<<endl;
14         cout<<"7.- 0: Pegado Opaco 1: Pegado Blending"<<endl;
15         return 0;
16     }
17     Imagen I, Ip;
18     I.LeerImagen(argv[1]);
19     Ip.LeerImagen(argv[2],argv[3]);
20     int i,j;
21     i=atoi(argv[5]); j=atoi(argv[6]);
22     Tipo_Pegado tp=OPACO;
23     int au= atoi(argv[7]);
24     if (au!=0)
25         tp=BLENDING;
26     I.PutImagen(i,j,Ip,tp);
27     I.EscribirImagen(argv[4]);
28 }
```





# Tarea 3.- Rutas Areas

Ruta: **Afganistan Alemania Camerun Ecuador España**





# Tarea 3.- Rutas Areas

Ruta: **Afganistan** Alemania Camerun Ecuador España





# Tarea 3.- Rutas Areas

Ruta: **Afganistan Alemania** Camerun Ecuador España





# Tarea 3.- Rutas Areas

Ruta: **Afganistan Alemania Camerun Ecuador España**





# Tarea 3.- Rutas Areas

Ruta: **Afganistan Alemania Camerun Ecuador España**

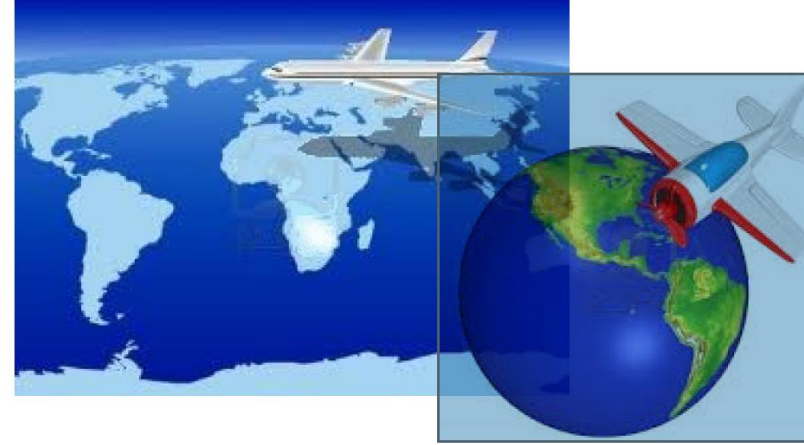




# Tarea 3.- Rutas Areas

Ruta: **Afganistan Alemania Camerun Ecuador España**

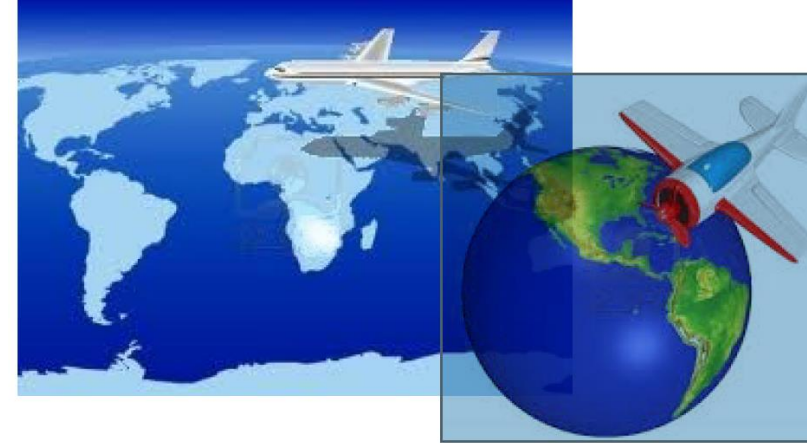




## Tarea 3.- Rutas Areas

- Subtareas a realizar.-

1. Leer Imagen del mapa
2. Leer imagen del avión y su máscara
3. Leer el conjunto de rutas (se carga en un almacén de rutas)
4. Mostrar las rutas y pedirle al usuario que escoja una (pedir un código)
5. Pintar el recorrido de la ruta en el mapa pintando las banderas de los países por donde pasa y un avión. Entre dos países se pinta el avión al principio, final y en el punto medio.



# Tarea 3.- Rutas Areas

Ejemplos de rutas (archivo almacen\_rutas.txt)

#Rutas

**R1** 5 (34.520418555522845,69.200820900000005) (52.50786264022465,13.426141949999987)  
(7.406652727545182,12.344585699999925) (-0.18659558628491132,-78.4305382) (40.40051528912146 ,-3.5916460749999635)

**R2** 8 (58.695433501291085,-96) (35.08690549340541,-103.72339606166992) (-12.055345316962327,-  
77.045185300000001) (40.40051528912146,-3.5916460749999635) (37.943768420529985,104.136111750000005) (-  
27.787075486256633,133.28132295) (35.673473752079516,139.710388000000008) (62.88  
647107195116,61.551173617626986)

**R3** 5 (17.246400332673307, -19.670602940234403) (4.283635422564345,-74.224039950000002)  
(51.528868434293244, -0.10159864999991441) (62.88647107195116,61.551173617626986)  
(37.943768420529985,104.136111750000005)

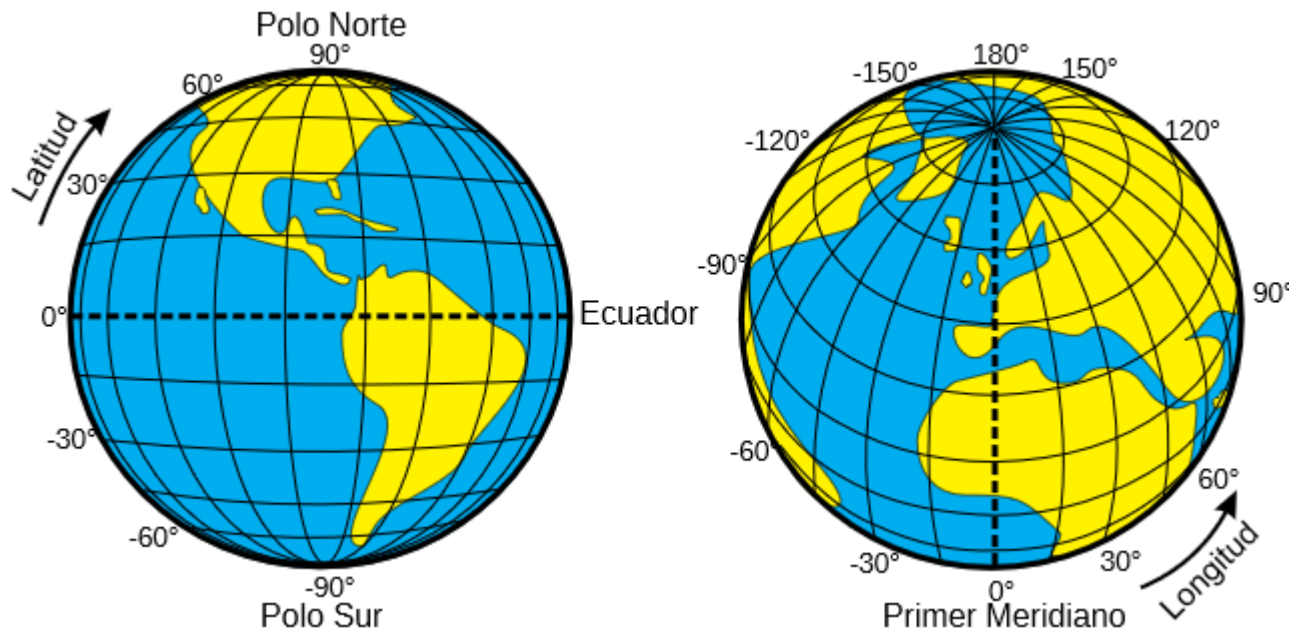
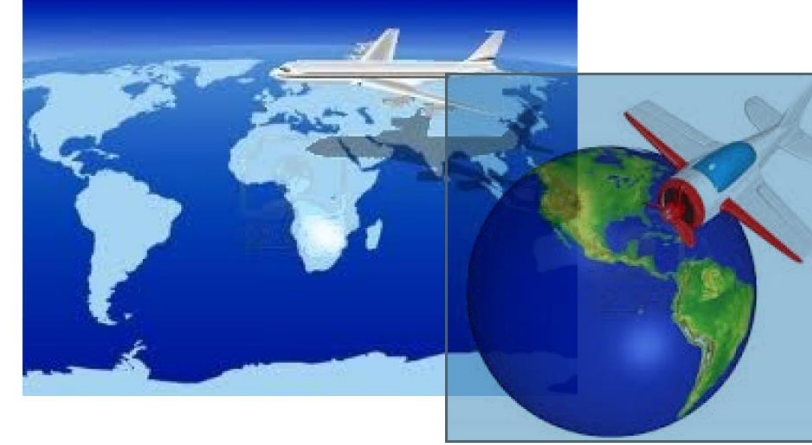
**R4** 11 (14.422538164676899,-87.63432239999997) (48.85887766623369,2.3470598999999766) (24.725939314861463  
,46.8225287999999986) (58.695433501291085,-96) (35.08690549340541,-103.72339606166992) (-12.055345316962327,-  
77.045185300000001) (40.40051528912146,-3.5916460749999635) (37.943768420529985,104.136111750000005) (-  
27.787075486256633,133.28132295) (35.673473752079516,139.710388000000008)  
(62.88647107195116,61.551173617626986)

**R5** 5 (52.76081718996433,8.747611999999986) (-19.051901092806112,29.152801800000002)(-34.61590069251671,-  
58.433298449999995) (58.695433501291085,-96) (52.76081718996433,8.747611999999986)



# Tarea 3.- Rutas Areas

## Concepto de Latitud y Longitud.



Ejemplo España =(40.40051528912146 ,-3.5916460749999635)



# Tarea 3.- Rutas Areas

¿Qué necesitamos?

1. Módulo Imagen + función Rotar
2. Módulo Punto
3. Módulo Pais
4. Módulo Países
5. Módulo Ruta
6. Módulo Almacen\_Rutas
7. Como pasar de pares (latitud,longitud) a una posición de la imagen mapa.
8. Función Pintar Ruta

# Tarea 3.- Rutas Areas

## Módulo Punto



```
11 class Punto{
12     private:
13         double latitud,longitud;
14
15     public:
16         Punto(){
17             latitud=longitud=0;
18
19         }
20         Punto(double l,double L, const string & d):latitud(l),longitud(L){}
21         double GetLatitud() const;
22         double GetLongitud() const;
23         void SetLatitud(double l);
24         void SetLongitud(double l);
25         bool operator<(const Punto &p) const;
26         bool operator==(const Punto &p) const;
27         friend istream & operator>>(istream & is, Punto &p);
28         friend ostream & operator<<(ostream & os, const Punto &p);
29     };
```

# Tarea 3.- Rutas Areas

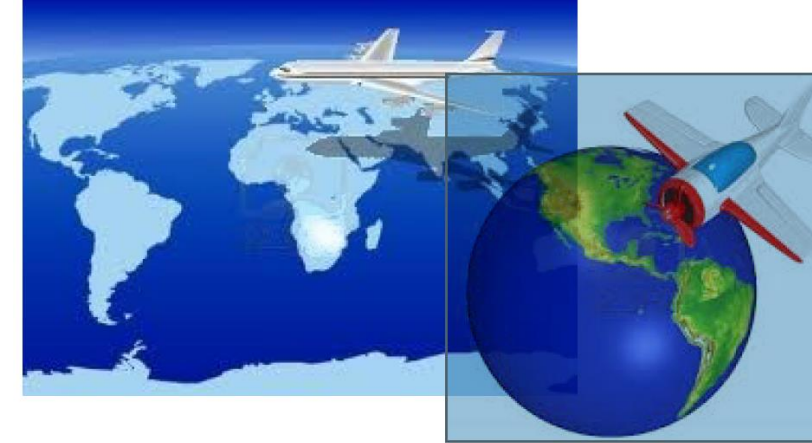
## Módulo País

```
3  #include "Punto.h"
4  class Pais{
5      private:
6          Punto p;
7          string pais;
8          string bandera;
9
10     public:
11         Pais() {}
12         Punto GetPunto() const;
13         string GetPais() const;
14         string GetBandera() const;
15         bool operator<(const Pais &P) const;
16         bool operator==(const Pais &P) const;
17         bool operator==(const Punto &P) const;
18         friend istream & operator>>(istream & is, Pais & P);
19         friend ostream & operator<<(ostream & os, const Pais &P)
20     };
21     ..
```



# Tarea 3.- Rutas Areas

## Módulo Países



```
4  #include "Pais.h"
5  using namespace std;
6  class Países{
7  private:
8      set<Pais> datos;
9  public:
10     Países();
11     void Insertar(const Pais &P);
12     void Borrar(const Pais &P);
13     class iterator{
14     private:
15         set<Pais>::iterator p;
16     public:
17         ...
18     };
19     class const_iterator{
20     private:
21         set<Pais>::const_iterator p;
22     public:
23         ...
24     };
30
31
32
33
34
35
36
37
38
39
40 };
```

```
iterator begin();
const_iterator begin() const;
iterator end();
const_iterator end() const;

iterator find(const Pais &p);
iterator find(const Punto &p);

friend istream & operator>>(istream & is, Países & R);
friend ostream & operator<<(ostream & os, const Países &R)
```



# Tarea 3.- Rutas Areas

## Módulo Ruta



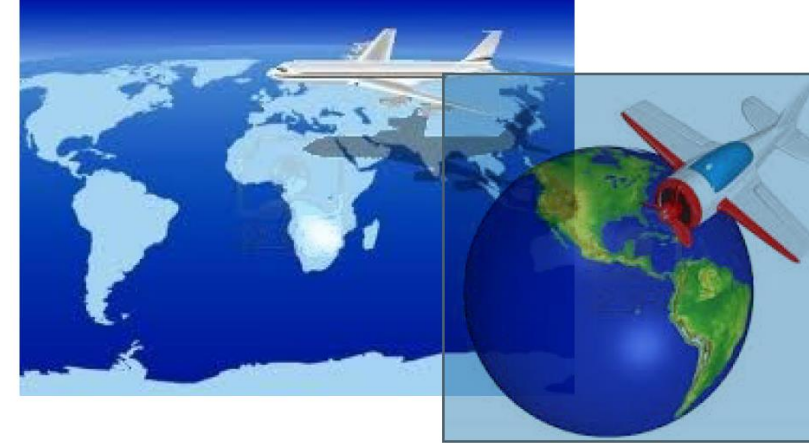
```
4  #include <string>
5  #include "Punto.h"
6  using namespace std;
7  class Ruta{
8  private:
9      list<Punto> puntos;
10     string code;
11 public:
12     Ruta();
13     void Insertar(const Punto & n);
14     void Borrar(const Punto &n);
15     string GetCode() const;
16
17     void SetCode(const string & code);
18     bool operator==(const Ruta &R) const;
19     bool operator<(const Ruta &R) const;
20 }
```

```
21 class iterator{
22 private:
23     list<Punto>::iterator p;
24 public:
25     ...
26 };
27 class const_iterator{
28 private:
29     list<Punto>::const_iterator p;
30 public:
31     ...
32 };
33 iterator begin();
34 const_iterator begin() const;
35 iterator end();
36 const_iterator end() const;
37
38 iterator find(const Punto &p);
39
40 friend istream & operator>>(istream & is, Ruta & R);
41 friend ostream & operator<<(ostream & os, const Ruta &R);
42 }
```

# Tarea 3.- Rutas Areas

## Módulo Almacen\_Rutas

```
1  #include <utility>
2
3  #include "Ruta.h"
4
5  class Almacen_Rutas{
6  private:
7      map<string, Ruta> rutas; //codigo de ruta y ruta
8  public:
9      Almacen_Rutas();
10     void Insertar(const Ruta & R);
11     void Borrar(const Ruta &R);
12     Ruta GetRuta(const string & a);
13     class iterator{
14     private:
15         map<string, Ruta>::iterator p;
16     public:
17         ...
18     };
19     class const_iterator{
20     private:
21         map<string, Ruta>::const_iterator p;
22     public:
23         ...
24     };
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40     iterator begin();
41     const_iterator begin() const;
42     iterator end();
43     const_iterator end() const;
44     friend istream & operator>>(istream & is, Almacen_Rutas & AR);
45     friend ostream & operator<<(ostream & os, const Almacen_Rutas &R);
46 }
```





# Tarea 3.- Rutas Areas



- Como pasar de pares (latitud,longitud) a una posición de la imagen mapa.

$$\begin{aligned} \text{columna} &= ( \text{totalcolumnas}/360.0 ) * (180 + \text{longitud}) \\ \text{fila} &= ( \text{totalfilas}/180.0 ) * (90 - \text{latitud} ) \end{aligned}$$

# Tarea 3.- Rutas Areas

## Pintar Ruta

1. Por cada par de puntos  $p1$  y el siguiente  $p2$  de la ruta
  1. Buscar el país  $c1$  que se corresponde con el punto  $p1$
  2. Buscar el país  $c2$  que se corresponde con el punto  $p2$
  3. Leer la imagen de la bandera del país  $c1$
  4. Leer la imagen de la bandera del país  $c2$
  5. Obtener la posición en el mapa de  $p1$  sea  $pos1\_mapa$
  6. Obtener la posición en el mapa de  $p2$  sea  $pos2\_mapa$
  7. Pegar el avión en el mapa en las posiciones  $pos1\_mapa$ ,  $pos2\_mapa$  y punto medio de  $pos1\_mapa$ ,  $pos2\_mapa$ . Antes de pegar rotar el avión según la línea que une  $pos1\_mapa$ ,  $pos2\_mapa$ .
  8. Pegar la bandera del país  $c1$  en  $pos1\_mapa$
  9. Pegar la bandera del país  $c2$  en  $pos2\_mapa$

