

# Envolvente convexa

Javier Goizueta

Mayo 1997

Este módulo de MapBasic calcula la *envolvente convexa* de una nube de puntos (función `ConvexHull`). La envolvente convexa es un polígono convexo que contiene a todos los puntos y cuyos vértices son puntos del conjunto; si colocamos un clavo en cada punto y a su alrededor una goma elástica que quede tensa, la forma que esta goma adopta es la de la envolvente convexa.

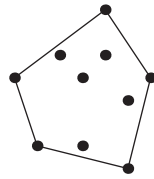


Figure 1: envolvente convexa

Una aplicación útil de esta función es el cálculo de “áreas percentiles”, áreas que contienen una proporción determinada de un conjunto de puntos cercana a un punto dado. Esta función está implementada por `Nearest`. También se define una forma de acceso interactiva a esta función mediante `NearDlg` que toma los datos de un cuadro de diálogo.

## 1 Programa

Estructura general del módulo: archivo `env.def` que declara el *interface*, y archivo `env.mb` con la implementación.

```
"env.def" 1a ≡
  <Declaraciones Públicas 4b, ... >
  ◇

"env.mb" 1b ≡
  <Interfaces Usados 1c, ... >
  <Definiciones 2a >
  <Declaraciones 2b, ... >
  <Implementación 3a, ... >
  ◇
```

Como en la mayoría de los programas de MapBasic, utilizaremos definiciones básicas de `mapbasic.def`.

```
<Interfaces Usados 1c > ≡
  Include "mapbasic.def"
  ◇
```

Macro defined by [1cd](#).  
Macro referenced in [1b](#).

También incluimos el interface propio, para declarar las funciones públicas a implementar.

```
<Interfaces Usados 1d > ≡
  Include "env.def"
  ◇
```

Macro defined by [1cd](#).  
Macro referenced in [1b](#).

## 1.1 Trigonometría

En primer lugar se precisan algunas funciones trigonométricas. Tendremos que usar frecuentemente el valor de la constante  $\pi$ , para ello definimos `Pi`. También definimos el identificador `HalfPi` con el valor  $\frac{\pi}{2}$  por comodidad y eficiencia:

```
<Definiciones 2a> ≡
  Define Pi 3.14159265358979323846
  Define HalfPi 1.57079632679489661923
  ◇
```

Macro referenced in [1b](#).

Defines: `HalfPi` [3a](#), `Pi` [3a](#), [6b](#), [7a](#).

La función `ATan2` es equivalente a las funciones `atan2` de Fortran o C; calcula el ángulo (argumento) de un vector (equivalentemente, de un número complejo). El valor devuelto es un ángulo en radianes con valor en  $[-\pi, \pi]$ .

En primer lugar, la declaración, en el lugar apropiado.

```
<Declaraciones 2b> ≡
  Declare Function ATan2(ByVal y As Float, ByVal x As Float) As Float
  ◇
```

Macro defined by [2b](#), [3b](#), [4a](#), [6d](#), [7b](#).

Macro referenced in [1b](#).

Uses: `ATan2` [3a](#).

Y ahora la definición; cuando el vector es  $(0, 0)$ , el valor de `ATan2` es indeterminado. Esta función no genera ningún tipo de error; en tal caso devuelve el valor de la variable (global visible sólo dentro de este módulo) `ATan2_Indet`, que por defecto es 0.

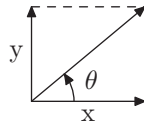


Figure 2: ángulo de un vector

```

<Implementación 3a> ≡
  Dim ATan2_Indet As Float
  Function ATan2(ByVal y As Float, ByVal x As Float) As Float
    Dim xsgn, ysgn As Integer
    xsgn = Sgn(x)
    ysgn = Sgn(y)
    Dim a As Float
    a = ATan2_Indet
    If (xsgn<>0 Or ysgn<>0) Then
      x = Abs(x)
      y = Abs(y)
      If (y>x) Then
        a = HalfPi - ATn(x/y)
      Else
        a = ATn(y/x)
      End If
      If (xsgn<0) Then a = Pi-a End If
      If (ysgn<0) Then a = -a End If
    End If
    ATan2 = a
  End Function
  ◇

```

Macro defined by [3ac](#), [5a](#), [6c](#), [7ac](#), [8b](#), [9b](#).  
 Macro referenced in [1b](#).  
 Defines: [ATan2 2b](#), [3c](#), [ATan2\\_Indet 6b](#).  
 Uses: [HalfPi 2a](#), [Pi 2a](#).

La función `AngleFromTo` calcula el valor del ángulo que va de un vector  $(x_1, y_1)$  a otro  $(x_2, y_2)$

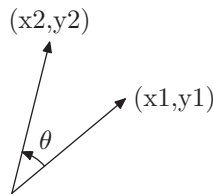


Figure 3: ángulo entre dos vectores

```

<Declaraciones 3b> ≡
  Declare Function AngleFromTo(
    ByVal x1 As Float, ByVal y1 As Float,
    ByVal x2 As Float, ByVal y2 As Float
  ) As Float
  ◇

```

Macro defined by [2b](#), [3b](#), [4a](#), [6d](#), [7b](#).  
 Macro referenced in [1b](#).  
 Uses: [AngleFromTo 3c](#).

```

<Implementación 3c> ≡
  Function AngleFromTo(
    ByVal x1 As Float, ByVal y1 As Float,
    ByVal x2 As Float, ByVal y2 As Float) As Float
    AngleFromTo = ATan2(y2*x1-x2*y1,x2*x1+y2*y1)
  End Function
  ◇

```

Macro defined by [3ac](#), [5a](#), [6c](#), [7ac](#), [8b](#), [9b](#).  
 Macro referenced in [1b](#).  
 Defines: [AngleFromTo 3b](#), [7a](#).  
 Uses: [ATan2 3a](#).

## 1.2 Algoritmo

El algoritmo usado para calcular la envolvente convexa de un conjunto de puntos  $(x_n, y_n)$  es el siguiente:

- En primer lugar se calcula un punto que será interior a la envolvente; el centroide o media  $(x_c, y_c)$  de los puntos.
- A continuación se toma el punto  $(x_{i0}, y_{i0})$  más alejado del centroide; este punto pertenece a la envolvente, y partiremos de él para construirla.
- Una vez tenemos un punto  $(x_{ij}, y_{ij})$  de la envolvente, el siguiente punto  $(x_{ij+1}, y_{ij+1})$  será aquél que maximice o minimice el valor de  $\alpha$ . (maximizando se recorre la envolvente en el sentido de las agujas del reloj —el interior queda a la derecha del perímetro; minimizando se invierte el sentido)

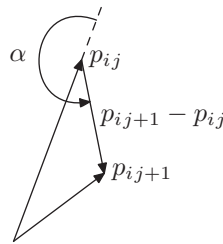


Figure 4: parámetro  $\alpha$

- El algoritmo finaliza una vez que se vuelve a encontrar el punto inicial.

El conjunto de puntos es pasado a la función como una tabla.

El parámetro  $\alpha$  a minimizar depende del centroide, del último punto de la envolvente,  $(x_{ij}, y_{ij})$ , y de cada punto a considerar. Para evitar pasar un gran número de argumentos y simplificar las expresiones, definiremos una función auxiliar que permita fijar el centroide y el primer punto en variables globales (visibles sólo en el módulo)

```
<Declaraciones 4a> ≡
  Dim Alpha_xc, Alpha_yc As Float ' centro de la envolvente
  Dim Alpha_x1, Alpha_y1 As Float ' primer vértice
  Declare Function Alpha(ByVal x As Float, ByVal y As Float) As Float
  ◇
```

Macro defined by [2b](#), [3b](#), [4a](#), [6d](#), [7b](#).  
 Macro referenced in [1b](#).  
 Defines: [Alpha\\_x1 5a, 6c](#), [Alpha\\_xc 5a, 6c](#), [Alpha\\_y1 5a, 6c](#), [Alpha\\_yc 5a, 6c](#).  
 Uses: [Alpha 6c](#).

```
<Declaraciones Públicas 4b> ≡
  Declare Sub ConvexHull(ByVal point_tab As String, env As Object)
  ◇
```

Macro defined by [4b](#), [8a](#), [9a](#).  
 Macro referenced in [1a](#).  
 Uses: [ConvexHull 5a](#).

```

<Implementación 5a> ≡
Sub ConvexHull(ByVal point_tab As String, env As Object)
  Dim xc,yc As Float
  <Calcular el centroide de point_tab en xc,yc 5b>

  Create Region Into Variable env 1 0
    Center (xc,yc) ' por desgracia esto es ignorado

  Dim x0, y0, xn, yn As Float
  <Calcular el primer vértice de la envolvente en x0,y0 6a>

  ' Fijamos el centroide para Alpha
  Alpha_xc = xc
  Alpha_yc = yc

  Dim n As SmallInt ' contador de vértices

  ' vértice inicial
  n = 1
  Print Str$(n)+" vértice"
  xn = x0
  yn = y0
  Alter Object env Node Add (xn,yn)

  Do
    ' Fijamos el punto inicial para Alpha
    Alpha_x1 = xn
    Alpha_y1 = yn

    ' vértice siguiente
    n = n+1
    Print Str$(n)+" vértices"
    <Calcular el punto de Point_Tab que minimice Alpha en xn,yn 6b>
    Alter Object env Node Add (xn,yn)

  Loop Until(xn=x0 And yn=y0)

  ' Forzamos a que se recalcule el centroide, ya que si no queda
  ' como uno de los vértices introducidos
  env = ConvertToRegion(ConvertToPLine(env))
  ' también podríamos haber hecho: env = OverlayNodes(env,env)

End Sub
◇

```

Macro defined by [3ac](#), [5a](#), [6c](#), [7ac](#), [8b](#), [9b](#).

Macro referenced in [1b](#).

Defines: ConvexHull [4b](#), [8b](#).

Uses: Alpha [6c](#), Alpha\_x1 [4a](#), Alpha\_xc [4a](#), Alpha\_y1 [4a](#), Alpha\_yc [4a](#).

```

<Calcular el centroide de point_tab en xc,yc 5b> ≡
  Dim n_points As Integer
  n_points = TableInfo(point_tab,TAB_INFO_NROWS)
  Select Sum(CentroidX(obj)) "X", Sum(CentroidY(obj)) "Y"
  From point_tab Into Tmp_Env
  Fetch First From point_tab
  xc = Tmp_Env.X / n_points
  yc = Tmp_Env.Y / n_points
  Close Table Tmp_Env
◇

```

Macro referenced in [5a](#).

```

< Calcular el primer vértice de la envolvente en x0 , y0 6a > ≡
Select
  Distance(xc,yc,CentroidX(obj),CentroidY(obj),"m"),
  CentroidX(obj) "X", CentroidY(obj) "Y"
From Point_Tab
Order By coll Desc
Into Tmp_Env
Fetch First From Tmp_Env
x0 = Tmp_Env.X
y0 = Tmp_Env.Y
Close Table Tmp_Env

```

◇

Macro referenced in 5a.

Al minimizar  $\alpha$  hay que tener en cuenta que entre los puntos Point\_Tab se encontrará el punto inicial Alpha\_x1, Alpha\_y1. Si la desigualdad de la función AngleAlpha no hubiera sido fijada correctamente de acuerdo con la operación a realizar, habría que establecer el valor ATan2\_Indet como Pi en caso de minimizar o como 0 si se va a maximizar  $\alpha$ .

```

< Calcular el punto de Point_Tab que minimice Alpha en xn, yn 6b > ≡
' ATan2_Indet = Pi ' valor para minimizar Alpha

```

```

Select
  Alpha(CentroidX(obj),CentroidY(obj)),
  CentroidX(obj) "X", CentroidY(obj) "Y"
From Point_Tab
Order By Coll Asc
Into Tmp_Env
Fetch First From Tmp_Env
xn = Tmp_Env.X
yn = Tmp_Env.Y
Close Table Tmp_Env

```

◇

Macro referenced in 5a.

Uses: Alpha 6c, ATan2\_Indet 3a, Pi 2a.

Función que calcula parámetro a maximizar, fijados el punto central Alpha\_xc, Alpha\_yc y el punto inicial Alpha\_x1, Alpha\_y1.

```

< Implementación 6c > ≡
Function Alpha(ByVal x As Float, ByVal y As Float) As Float
  Alpha = AngleAlpha(Alpha_x1-Alpha_xc,Alpha_y1-Alpha_yc,
                    x-Alpha_xc,y-Alpha_yc)
End Function

```

◇

Macro defined by 3ac, 5a, 6c, 7ac, 8b, 9b.

Macro referenced in 1b.

Defines: Alpha 4a, 5a, 6b, 7a.

Uses: Alpha\_x1 4a, Alpha\_xc 4a, Alpha\_y1 4a, Alpha\_yc 4a, AngleAlpha 7a.

Ángulo  $\alpha$  a minimizar. (ver figura 4, página 4)

```

< Declaraciones 6d > ≡
Declare Function AngleAlpha(
  ByVal x1 As Float, ByVal y1 As Float,
  ByVal x2 As Float, ByVal y2 As Float) As Float

```

◇

Macro defined by 2b, 3b, 4a, 6d, 7b.

Macro referenced in 1b.

Uses: AngleAlpha 7a.

```

<Implementación 7a> ≡
Function AngleAlpha(ByVal x1 As Float, ByVal y1 As Float,
                   ByVal x2 As Float, ByVal y2 As Float) As Float
    x2 = x2-x1
    y2 = y2-y1
    Dim a As Float
    a = AngleFromTo(x1,y1,x2,y2)
    If a<=0 Then ' cambiar por a<0 si se va a maximizar Alpha
        a = a+2*Pi
    End If
    AngleAlpha = a
End Function
◇

```

Macro defined by [3ac](#), [5a](#), [6c](#), [7ac](#), [8b](#), [9b](#).  
 Macro referenced in [1b](#).  
 Defines: AngleAlpha [6cd](#).  
 Uses: Alpha [6c](#), AngleFromTo [3c](#), Pi [2a](#).

La función AuxColVal resulta muy útil para referirse a columnas de una tabla cuando el nombre de la tabla o de la columna se halla en una variable de texto.

Ésta función coincide con la función ColVal del módulo colval del programa RTP. Para independizar esté módulo de aquél programa la implemento también aquí, cambiando el nombre; cuando esté preparada mi biblioteca de utilidades MapBasic, será incluida allí.

```

<Declaraciones 7b> ≡
Declare Function AuxColVal(ByVal tab As String,
                          ByVal col As String) As Alias
◇

```

Macro defined by [2b](#), [3b](#), [4a](#), [6d](#), [7b](#).  
 Macro referenced in [1b](#).  
 Uses: AuxColVal [7c](#).

La implementación original, AuxColVal=tab+"."+col da problemas en algunas expresiones.

```

<Implementación 7c> ≡
Function AuxColVal(ByVal tab As String,
                  ByVal col As String) As Alias
    'AuxColVal = tab+"."+col ' esto da problemas en ciertos casos
    Dim a As Alias
    a = tab+"."+col
    AuxColVal = a
End Function
◇

```

Macro defined by [3ac](#), [5a](#), [6c](#), [7ac](#), [8b](#), [9b](#).  
 Macro referenced in [1b](#).  
 Defines: AuxColVal [7b](#), [9b](#), [11b](#).

### 1.3 Puntos cercanos

La función Nearest calcula el área que contiene a la fracción más cercana a (xc, yc) de los puntos de la tabla tabname.

Antes de usar esta función se debe establecer el sistema de coordenadas apropiado para (xc, yc) mediante una instrucción del tipo Set CoordSys . . .

```

<Declaraciones Públicas 8a> ≡
  Declare Sub Nearest
    (ByVal tabname As String,           ' tabla de puntos
     ByVal f As Float,                 ' fracción de puntos
     ByVal xc As Float, ByVal yc As Float, ' centro
     env As Object)                   ' resultado
  ◇

```

Macro defined by [4b](#), [8a](#), [9a](#).

Macro referenced in [1a](#).

Uses: Nearest [8b](#).

```

<Implementación 8b> ≡
  Sub Nearest(ByVal tabname As String,
              ByVal per As Float,
              ByVal xc As Float, ByVal yc As Float,
              env As Object)
    If per<1 Then
      Select Distance(xc,yc,CentroidX(obj),CentroidY(obj),"m")
        From tabname
        Order By Coll
        Into Tmp_Nearest0
      Dim nrows, threshold As Integer
      nrows = TableInfo(Tmp_Nearest0,TAB_INFO_NROWS)
      threshold = nrows*per
      Select * From Tmp_Nearest0
        Where RowID<=threshold Into Tmp_Nearest
    Else
      Select * From tabname Into Tmp_Nearest
    End If
    Call ConvexHull("Tmp_Nearest",env)
    Close Table "Tmp_Nearest"
    If per<1 Then
      Close Table "Tmp_Nearest0"
    End If
  End Sub
  ◇

```

Macro defined by [3ac](#), [5a](#), [6c](#), [7ac](#), [8b](#), [9b](#).

Macro referenced in [1b](#).

Defines: Nearest [8a](#), [11b](#).

Uses: ConvexHull [5a](#).

## 1.4 Interfaz de Usuario

La función NearDlg presenta un em interface interactivo para Nearest. La envolvente creada la coloca en la capa descriptiva de la ventana de mapa seleccionada.

El uso de esta función sería así:

- Si no se dispone de una tabla que contenga únicamente el punto central, seleccionar este punto (crearlo si es necesario en una capa descriptiva)
- Ejecutar la utilidad que ejecute NearDlg
- En centro elegir la tabla que contiene el centro (e.g. la selección)
- En puntos elegir la tabla de los puntos a envolver (si la tabla contiene elementos no puntuales se envolverán sus centroides)
- Introducir el porcentaje de puntos a envolver
- Elegir una ventana de mapa, en cuya capa descriptiva se colocará la envolvente generada.

⟨Declaraciones Públicas 9a⟩ ≡

Declare Sub NearDlg

◇

Macro defined by [4b](#), [8a](#), [9a](#).

Macro referenced in [1a](#).

Uses: NearDlg [9b](#).

⟨Implementación 9b⟩ ≡

Sub NearDlg

Dim i As SmallInt

' Vector de tablas mapificables

Dim maptabs(1) As String

Dim n\_maptabs As SmallInt

⟨Llena vector de tablas mapificables, n\_maptabs, maptabs [10a](#)⟩

' Vector de ventanas de mapa

Dim mapwins(1) As Integer

Dim maptitles As String

Dim n\_mapwins As SmallInt

⟨Llena el vector de ventanas de mapa, n\_mapwins, mapwins, maptitles [10b](#)⟩

' Valores seleccionables en el diálogo

Dim center\_i, points\_i, map\_i As SmallInt

Dim percent As Float

⟨Establece valores iniciales de center\_i, points\_i, maps\_i [11a](#)⟩

' establece fracción inicial (50%)

percent = 50

⟨Cuadro de diálogo [11c](#)⟩

If CommandInfo(CMD\_INFO\_DLG\_OK) Then

' Se debe haber pulsado OK...

If center\_i>0 And points\_i>0 And map\_i>0 Then

' ...con selecciones válidas...

' nombre de la tabla que contiene el punto central

Dim tabname As String

tabname = maptabs(center\_i)

If tabname="Selección" Then

tabname = "Selection"

End If

If TableInfo(tabname, TAB\_INFO\_NROWS)=1 Then

Fetch First From tabname

If Not AuxColVal(tabname, "Obj") Then

Note "El centro debe ser un único punto"

Else

⟨Crear área en el mapa map\_i con tabname, percent [11b](#)⟩

End If

Else

Note "El centro debe ser un único punto"

End If

End If

End If

End Sub

◇

Macro defined by [3ac](#), [5a](#), [6c](#), [7ac](#), [8b](#), [9b](#).

Macro referenced in [1b](#).

Defines: NearDlg [9a](#).

Uses: AuxColVal [7c](#).

```

< Llena vector de tablas mapificables, n_maptabs, maptabs 10a > ≡
    Dim n_tabs As SmallInt

    n_tabs = NumTables()
    ReDim maptabs(n_tabs+1)
    n_maptabs = 0

    ' Si hay una selección activa y mapificable, se establece
    ' como primera tabla, con el nombre "Selección"
    If SelectionInfo(SEL_INFO_NROWS)>0 Then
        If TableInfo(SelectionInfo(SEL_INFO_TABLENAME),TAB_INFO_MAPPABLE) Then
            n_maptabs = n_maptabs+1
            maptabs(n_maptabs) = "Selección"
        End If
    End If

    For i = 1 to n_tabs
        If TableInfo(i,TAB_INFO_MAPPABLE) Then
            n_maptabs = n_maptabs+1
            maptabs(n_maptabs) = tableinfo(i,TAB_INFO_NAME)
        End If
    Next

    ReDim maptabs(n_maptabs)

```

◇

Macro referenced in [9b](#).

```

< Llena el vector de ventanas de mapa, n_mapwins, mapwins, maptitles 10b > ≡
    Dim n_wins As SmallInt
    Dim win_id As Integer

    n_wins = NumWindows()
    ReDim mapwins(n_wins)
    n_mapwins = 0

    For i = 1 to n_wins
        win_id = WindowID(i)
        If WindowInfo(win_id,WIN_INFO_TYPE)=WIN_MAPPER Then
            n_mapwins = n_mapwins+1
            mapwins(n_mapwins) = win_id
            maptitles = maptitles + WindowInfo(win_id,WIN_INFO_NAME) + ";"
        End If
    Next
    ReDim mapwins(n_mapwins)

```

◇

Macro referenced in [9b](#).

```

<Establece valores iniciales de center_i, points_i, maps_i 11a> ≡
' centro: selección o primera tabla mapificable
center_i = Minimum(1,n_maptabs)

' puntos: primera tabla mapificable, no la selección
points_i = center_i
If maptabs(points_i)="Selección" Then
    points_i = Minimum(points_i+1,n_maptabs)
End If

' mapa: primera ventana de mapa
map_i = Minimum(1,n_mapwins)

```

Macro referenced in [9b](#).

```

<Crear área en el mapa map_i con tabname, percent 11b> ≡
' punto central; ---objeto de la tabla tabname
Dim cnt As Object
cnt = AuxColVal(tabname,"Obj")

' coordenadas del punto central
Dim cntx, cnty As Float
cntx = CentroidX(cnt)
cnty = CentroidY(cnt)

' fracción de los puntos; ---percent%
Dim fract As Float
fract = percent*0.01

' área buscada
Dim env As Object
Call Nearest(maptabs(points_i), fract, cntx, cnty, env)

' Insertamos el área en la capa descriptiva del mapa; ---ventana map_i
Dim restab As String
restab = LayerInfo(mapwins(map_i),0,LAYER_INFO_NAME)
Insert Into restab (Obj) Values (env)

```

Macro referenced in [9b](#).

Uses: [AuxColVal 7c](#), [Nearest 8b](#).

El cuadro de diálogo emplea las variables maptabs, maptitles, valores center\_i, points\_i, percent y map\_i.

```

<Cuadro de diálogo 11c> ≡
Dialog
    Title "Área Envolvente"
    Width 178 Height 94

    Control StaticText
        Title "Centro"
        Position 5,11
    Control PopUpMenu
        Value center_i
        Into center_i
        Title From Variable maptabs
        Position 60,10 Width 115 'Height 12

    Control StaticText
        Title "Puntos"
        Position 5,26

```

```

Control PopUpMenu
  Value points_i
  Into points_i
  Title From Variable maptabs
  Position 60,25 Width 115 'Height 12

Control StaticText
  Title "Fracción de los puntos (%)"
  Position 5,41
Control EditText
  Value percent
  Into percent
  Position 95,40 Width 80 'Height 12

Control StaticText
  Title "Resultado en"
  Position 5,56
Control PopUpMenu
  Value map_i
  Into map_i
  Title From Variable maptitles
  Position 60,55 Width 115 'Height 12

Control OKButton Title "&Aceptar"
  Position 85,74
Control CancelButton Title "&Cancelar"
  Position 130,74

```

◇

Macro referenced in [9b](#).

## 2 Índices

### 2.1 Archivos

"env.def" Defined by [1a](#).

"env.mb" Defined by [1b](#).

### 2.2 Fragmentos

⟨ Calcular el centroide de point\_tab en xc , yc [5b](#) ⟩ Referenced in [5a](#).

⟨ Calcular el primer vértice de la envolvente en x0 , y0 [6a](#) ⟩ Referenced in [5a](#).

⟨ Calcular el punto de Point\_Tab que minimice Alpha en xn , yn [6b](#) ⟩ Referenced in [5a](#).

⟨ Crear área en el mapa map\_i con tabname, percent [11b](#) ⟩ Referenced in [9b](#).

⟨ Cuadro de diálogo [11c](#) ⟩ Referenced in [9b](#).

⟨ Declaraciones Públicas [4b](#), [8a](#), [9a](#) ⟩ Referenced in [1a](#).

⟨ Declaraciones [2b](#), [3b](#), [4a](#), [6d](#), [7b](#) ⟩ Referenced in [1b](#).

⟨ Definiciones [2a](#) ⟩ Referenced in [1b](#).

⟨ Establece valores iniciales de center\_i, points\_i, maps\_i [11a](#) ⟩ Referenced in [9b](#).

⟨ Implementación [3ac](#), [5a](#), [6c](#), [7ac](#), [8b](#), [9b](#) ⟩ Referenced in [1b](#).

⟨ Interfaces Usados [1cd](#) ⟩ Referenced in [1b](#).

⟨ Llena el vector de ventanas de mapa, n\_mapwins, mapwins, maptitles [10b](#) ⟩ Referenced in [9b](#).

⟨ Llena vector de tablas mapificables, n\_maptabs, maptabs [10a](#) ⟩ Referenced in [9b](#).

### 2.3 Identificadores

Alpha: [4a](#), [5a](#), [6b](#), [6c](#), [7a](#).

Alpha\_x1: [4a](#), [5a](#), [6c](#).

Alpha\_xc: [4a](#), [5a](#), [6c](#).

Alpha\_y1: [4a](#), [5a](#), [6c](#).

Alpha\_yc: [4a](#), [5a](#), [6c](#).

AngleAlpha: [6cd](#), [7a](#).

AngleFromTo: [3b](#), [3c](#), [7a](#).

ATan2: [2b](#), [3a](#), [3c](#).  
ATan2\_Indet: [3a](#), [6b](#).  
AuxColVal: [7b](#), [7c](#), [9b](#), [11b](#).  
ConvexHull: [4b](#), [5a](#), [8b](#).  
HalfPi: [2a](#), [3a](#).  
NearDlg: [9a](#), [9b](#).  
Nearest: [8a](#), [8b](#), [11b](#).  
Pi: [2a](#), [3a](#), [6b](#), [7a](#).