PMCLIB Library User Guide

INTRODUCCIÓN

La librería PMCLIB contiene una serie de estructuras de datos, funciones y procedimientos básicos que permiten facilitar el trabajo con los ficheros estructurales de proteínas contenidos en la base de datos “PDB” (“Protein Data Bank”).

FICHEROS DE ESTRUCTURA DEL PROTEIN DATA BANK (rcsb.org)

La base de datos PDB (“Protein Data Bank”; https://www.rcsb.org/) agrupa 201789 estructuras de proteínas en formato “PDB”, calculadas experimentalmente mediante difractometría de rayos X de cristales de proteína o mediante resonancia magnética nuclear de muestras en disolución (dato actualizado a 28-02-2023). Asimismo, desde el año 2017 se han incorporado 1068577 estructuras calculadas computacionalmente por el algoritmo de Inteligencia Artificial “Alphafold” (DeepMind de Alphabet/Google). Estas últimas han sido recogidas en formato “CSM” (“Computed Structure Models”). Ambos formatos (PDB y CSM) se organizan como ficheros textuales independientes para cada proteína. Este último puede también obtenerse en formato PDB, por ejemplo, a partir de la base de datos “Alphafold DB”, accesible desde el propio portal de RCSB. En la página siguiente se reproduce un fragmento de fichero PDB en el que se aprecian sus dos características fundamentales: son ficheros de texto (que pueden ser leídos con cualquier tratador de textos) y están organizados por líneas encabezadas por una palabra clave que indica el tipo de información que sigue.

Entre estas líneas se incluyen las encabezadas por la palabra ATOM, que contienen los datos estructurales de cada átomo de la proteína. La información de estas líneas está rígidamente encolumnada, de modo que, a efectos de desarrollar algoritmos que puedan leer esta información, se deberá atender a las columnas que han sido especificadas en el formato PDB. La figura 2 muestra el tipo de información contenida en los registros ATOM y su encolumnado correspondiente.

Los átomos de un mismo residuo (aminoácido) están también agrupados en un orden preciso. Se empieza por los átomos N, CA, C y O que afectan al eje covalente de la proteína y a continuación se sitúan los restantes átomos de la cadena lateral del residuo, identificados con la nomenclatura específica del formato PDB. Esta disposición, como se verá más adelante, es importante a la hora de desarrollar “wrapers” capaces de rastrear la información del fichero PDB.

```
HEADER    OXIDOREDUCTASE                          25-AUG-95   1AHI
TITLE     7 ALPHA-HYDROXYSTEROID DEHYDROGENASE COMPLEXED WITH NADH
TITLE    2 AND 7-OXO GLYCOCHENODEOXYCHOLIC ACID
COMPND    MOL_ID: 1;
COMPND   2 MOLECULE: 7 ALPHA-HYDROXYSTEROID DEHYDROGENASE;
COMPND   3 CHAIN: A, B;
COMPND   4 EC: 1.1.1.159;
COMPND   5 ENGINEERED: YES
SOURCE    MOL_ID: 1;
SOURCE   2 ORGANISM_SCIENTIFIC: ESCHERICHIA COLI;
SOURCE   3 STRAIN: HB101;
SOURCE   4 EXPRESSION_SYSTEM: ESCHERICHIA COLI;
SOURCE   5 EXPRESSION_SYSTEM_STRAIN: DH1
KEYWDS    OXIDOREDUCTASE, SHORT-CHAIN DEHYDROGENASE/REDUCTASE,
EXPDTA    X-RAY DIFFRACTION
AUTHOR    N.TANAKA,T.NONAKA,Y.MITSUI
REVDAT   1   14-OCT-96 1AHI    0
JRNL        AUTH   N.TANAKA,T.NONAKA,T.TANABE,T.YOSHIMOTO,D.TSURU,
JRNL        AUTH 2 Y.MITSUI
JRNL        TITL   CRYSTAL STRUCTURES OF THE BINARY AND TERNARY
JRNL        TITL 2 COMPLEXES OF 7 ALPHA-HYDROXYSTEROID DEHYDROGENASE
JRNL        TITL 3 FROM ESCHERICHIA COLI
JRNL        REF    BIOCHEMISTRY                V.  35  7715 1996
JRNL        REFN   ASTM BICHAW  US ISSN 0006-2960                0033
REMARK   1
REMARK   1 REFERENCE 1
REMARK   1  AUTH   N.TANAKA,T.NONAKA,T.YOSHIMOTO,D.TSURU,Y.MITSUI
REMARK   1  TITL   CRYSTALLIZATION AND PRELIMINARY X-RAY
REMARK   1  TITL 2 CRYSTALLOGRAPHIC STUDIES OF 7ALPHA-HYDROXYSTEROID
REMARK   1  TITL 3 DEHYDROGENASE FROM ESCHERICHIA COLI
REMARK   1  REF    TO BE PUBLISHED

  .
  .
  .
  .
  .
ATOM      1  N   MET A   1      22.555  17.616  31.228  1.00 36.12           N
ATOM      2  CA  MET A   1      23.508  17.733  32.355  1.00 36.38           C
ATOM      3  C   MET A   1      24.913  17.439  31.862  1.00 32.79           C
ATOM      4  O   MET A   1      25.145  16.443  31.202  1.00 34.89           O
ATOM      5  CB  MET A   1      23.127  16.754  33.451  1.00 43.00           C
ATOM      6  CG  MET A   1      23.948  16.891  34.708  1.00 51.55           C
ATOM      7  SD  MET A   1      23.266  15.913  36.051  1.00 58.65           S
ATOM      8  CE  MET A   1      21.563  16.449  36.030  1.00 58.77           C
ATOM      9  N   PHE A   2      25.834  18.339  32.149  1.00 32.21           N
ATOM     10  CA  PHE A   2      27.219  18.192  31.737  1.00 34.36           C
ATOM     11  C   PHE A   2      27.829  16.939  32.353  1.00 35.52           C
ATOM     12  O   PHE A   2      27.511  16.600  33.491  1.00 44.20           O
ATOM     13  CB  PHE A   2      27.984  19.434  32.191  1.00 35.64           C
ATOM     14  CG  PHE A   2      29.473  19.320  32.087  1.00 34.59           C
ATOM     15  CD1 PHE A   2      30.121  19.565  30.889  1.00 33.78           C
ATOM     16  CD2 PHE A   2      30.234  19.013  33.206  1.00 35.34           C
```

Fig. 1.- Formato PDB

COLUMNS	DATA TYPE	FIELD	DEFINITION
1 - 6	Record name	"ATOM "	
7 - 11	Integer	serial	Atom serial number.
13 - 16	Atom	name	Atom name.
17	Character	altLoc	Alternate location indicator.
18 - 20	Residue name	resName	Residue name.
22	Character	chainID	Chain identifier.
23 - 26	Integer	resSeq	Residue sequence number.
27	AChar	iCode	Code for insertion of residues.
31 - 38	Real(8.3)	x	Orthogonal coordinates for X in Angstroms.
39 - 46	Real(8.3)	y	Orthogonal coordinates for Y in Angstroms.
47 - 54	Real(8.3)	z	Orthogonal coordinates for Z in Angstroms.
55 - 60	Real(6.2)	occupancy	Occupancy.
61 - 66	Real(6.2)	tempFactor	Temperature factor.
77 - 78	LString(2)	element	Element symbol, right-justified.
79 - 80	LString(2)	charge	Charge on the atom.

Fig. 2.- Formato PDB

JERARQUÍA DE CLASES DE LA LIBRERÍA

La librería PMCLIB contiene las Clases necesarias para representar las proteínas y sus componentes como estructuras de datos en una aplicación Free Pascal/Lázarus. Asimismo contiene una colección básica de métodos que permiten dotar a cada una de las clases creadas de la funcionalidad mínima necesaria para manipular estos objetos en la aplicación. Se trata de una librería en desarrollo aún, por lo que no pueden descartarse, en esta fase, la presencia de errores o inconsistencias. Una forma efectiva de corregirlas y de contribuir a su mejora es enviarme los errores, comentarios o sugerencias que estimes oportunos hilario@ugr.es. A continuación se describen las clases que componen la librería en este momento y los métodos incorporados.

Clase TTRANSFORM

```
TTransform = Class //-----------------------------------------------------
    private
      FPosition: TVector3;
      function  GetX: real;
      procedure SetX(valor: real);
      function  GetY: real;
      procedure SetY(valor: real);
      function  GetZ: real;
      procedure SetZ(valor: real);
    public
      function Module:  real;
      function Translate(dx, dy: real; dz: real = 0):    TTransform;
      function RotateX(radians: real):        TTransform;
      function RotateX(sinus, cosinus: real): TTransform;
      function RotateY(radians: real):        TTransform;
      function RotateY(sinus, cosinus: real): TTransform;
      function RotateZ(radians: real):        TTransform;
      function RotateZ(sinus, cosinus: real): TTransform;
      function Rotate(radX, radY: real; radZ: real = 0):     TTransform;
      function Rotate(sinusX, cosinusX, sinusY, cosinusY: real;
                      sinusZ: real = 0; cosinusZ: real = 1): TTransform;
      function Scale(factor: real):           TTransform;
      function Scale(factorX, factorY, factorZ: real):   TTransform;
      property X: real read GetX write SetX;
      property Y: real read GetY write SetY;
      property Z: real read GetZ write SetZ;
      property Position: TVector3 read FPosition write FPosition;
  end;
```

La clase TTRANSFORM está diseñada para dotar a sus descendientes de la capacidad de posicionarse y transformarse en el Espacio Afín. En este momento consta de un único campo privado: el registro denominado FPOSITION, de tipo TVector3, que representa el vector de posición al punto considerado. En el futuro, emulando a las estructuras equivalentes de algunos entornos comerciales de simulación y representación gráfica 3D (Unreal Engine, Unity y otros) dispondrá de otros dos registros análogos para representar el giro y el escalamiento y de la capacidad de implementar coordenadas locales y globales.
TTRANSFORM dispone además de una serie de métodos que le dan una funcionalidad básica a las instancias de esta clase y sus descendientes:

function Module: real.- Calcula el módulo del vector de posición a partir de las coordenadas consignadas en FPOSITION.

function Translate (dx, dy: real; dz: real = 0): TTransform.- Devuelve un registro de tipo TTransform con las coordenadas trasladadas. Dx, dy y dz son los incrementos aplicados en los ejes respectivos OX, OY y OZ. FPOSITION adopta los nuevos valores además de devolverlos.

function RotateX (radians: real): TTransform.- Lleva a cabo un giro sobre el eje OX del vector FPOSITION que, consecuentemente, resulta afectado. El nuevo valor, además, es devuelto como registro TTransform. “Radians” es el valor de rotación expresado en radianes.

function RotateX(sinus, cosinus: real): TTransform.- Sobrecarga de la función anterior en la que los argumentos “sinus” y “cosinus” se refieren, respectivamente, al seno y al coseno del ángulo de giro. Adecuado para evitar recalcular esos valores cuando se aplican a un elevado número de puntos. Lleva a cabo un giro sobre el eje OX del vector FPOSITION que, consecuentemente, resulta afectado. El nuevo valor, además, es devuelto como registro TTransform.

function RotateY(radians: real): TTransform.- Lleva a cabo un giro sobre el eje OY del vector FPOSITION que, consecuentemente, resulta afectado. El nuevo valor, además, es devuelto como registro TTransform. “Radians” es el valor de rotación expresado en radianes.

function RotateY(sinus, cosinus: real): TTransform.- Sobrecarga de la función anterior en la que los argumentos “sinus” y “cosinus” se refieren, respectivamente, al seno y al coseno del ángulo de giro. Adecuado para evitar recalcular esos valores cuando se aplican a un elevado número de puntos. Lleva a cabo un giro sobre el eje OY del vector FPOSITION que, consecuentemente, resulta afectado. El nuevo valor, además, es devuelto como registro TTransform.

function RotateZ(radians: real): TTransform.- Lleva a cabo un giro sobre el eje OZ del vector FPOSITION que, consecuentemente, resulta afectado. El nuevo valor, además, es devuelto como registro TTransform. “Radians” es el valor de rotación expresado en radianes.

function RotateZ(sinus, cosinus: real): TTransform.- Sobrecarga de la función anterior en la que los argumentos “sinus” y “cosinus” se refieren, respectivamente, al seno y al coseno del ángulo de giro. Adecuado para evitar recalcular esos valores cuando se aplican a un elevado número de puntos. Lleva a cabo un giro sobre el eje OZ del vector FPOSITION que, consecuentemente, resulta afectado. El nuevo valor, además, es devuelto como registro TTransform.

function Rotate(radX, radY: real; radZ: real = 0): TTransform.- Lleva a cabo un giro combinado sobre los ejes OX, OY y OZ del vector FPOSITION que, consecuentemente, resulta afectado. El nuevo valor, además, es devuelto como registro TTransform. “RadX”, “RadY” y “RadZ” son los valores de rotación sobre los ejes OX, OY y OZ expresados en radianes.

function Rotate(sinusX, cosinusX, sinusY, cosinusY: real; sinusZ: real = 0; cosinusZ: real = 1): TTransform.- Sobrecarga de la función anterior en la que los argumentos “sinusX”, “cosinusX”, “sinusY”, “cosinusY”, “sinusZ” y “cosinusZ” se refieren, respectivamente, a los senos y cosenos de los ángulos de giro sobre los ejes OX, OY y OZ. Adecuado para evitar recalcular esos valores cuando se aplican a un elevado número de puntos. Lleva a cabo un giro combinado sobre los ejes OX, OY y OZ del vector FPOSITION que, consecuentemente, resulta afectado. El nuevo valor, además, es devuelto como registro TTransform.

function Scale(factor: real): TTransform.- Lleva a cabo un escalamiento no deformante que aplica el mismo valor (factor) de escala a los ejes OX, OY y OZ. Como resultado el vector FPOSITION resulta modificado y el valor es devuelto como TTRANSFORM.

function Scale(factorX, factorY, factorZ: real): TTransform.- Lleva a cabo un escalamiento deformante que permite aplicar factores de escala diferentes (factorX, factory y factorZ) a cada uno de los ejes OX, OY y OZ del vector FPOSITION que, consecuentemente, resulta afectado. El resultado es además devuelto como TTRANSFORM.

property X: real read GetX write SetX.- Permite hacer público el valor privado de la coordenada X (FPOSITION.X).

property Y: real read GetY write SetY.- Permite hacer público el valor privado de la coordenada Y (FPOSITION.Y).

property Z: real read GetZ write SetZ.- Permite hacer público el valor privado de la coordenada Z (FPOSITION.Z).

property Position: TVector3 read FPosition write FPosition.- Permite acceder de forma pública al vector de posición FPosition de tipo TTRANSFORM.

Clase TATOM

```
TAtom = Class(TTransform) //-------------------------------------------------
  private
    FColor: TColor;
    FRadium: Real;
    FMarked: boolean;
  public
    constructor CreateAtom(cx: real = 0; cy: real = 0; cz: real = 0;
                cMarked: boolean = false; cradium: real = 1;
                ccolor: TColor = clwhite);
    property Color: TColor Read FColor  Write FColor;
    property Radium:  real Read FRadium Write FRadium;
    property marked: boolean Read FMarked Write FMarked;
end;
```

Clase descendiente de TTransform a la que añade tres campos privados (color, radio y marca), que se añaden al vector de posición heredado desde TTransform, y que son accesibles mediante las propiedades Color, Radium y Marked. Dispone además de un constructor (recomendado) que permite crear instancias iniciadas con valores específicos o por defecto.

constructor CreateAtom(cx: real = 0; cy: real = 0; cz: real = 0; cMarked: boolean = false; cradium: real = 1; ccolor: TColor = clwhite) .- Permite crear instancias de la clase TAtom con valores específicos de todos sus campos. También puede invocarse sin parámetros o solo con parte de ellos. Las instancias creadas con este constructor deben destruirse, cuando ya no se necesiten, con el destructor “Free”.

```
[Ejemplo]:          var
                      MiAtomo: TAtom;
                    …
                    …
                    begin
                      MIAtomo:= TAtom.CreateAtom (1,2,3);
                      …
                      MiAtomo.Free;
                    End;
```

crea una instancia de TAtom en la posición indicada, radio unidad, color blanco y marca "false".

property Color: TColor Read FColor Write FColor.- Permite acceder o modificar el valor del campo "color" del Átomo (FCOLOR).

property Radium: real Read FRadium Write FRadium.- Permite acceder o modificar el valor del campo "radio" del Átomo (FADIUM).

property Marked: boolean Read FMarked Write FMarked.- Permite acceder o modificar el valor del campo "marcado" del Átomo (FMARKED).

Clase TATOMPDB

```
  TAtomPDB = Class(TAtom) //-------------------------------------------------
-
    private
      FSerialNumber: integer;
      FId: string;
      FResidue: string;
      FSubunit: char;
      FResNumber: integer;
      FTemp: real;
    public
      constructor CreateAtomPDB(cx: real = 0; cy: real = 0; cz: real = 0;
                  cmarked: boolean = false; ccolor: TColor = clwhite;
                  cradium: real = 1; cSerialNumber: integer = 0; cId: string =
'NA';
                  cresidue: string = 'NA'; cSubunit: char = ' ';
                  cResNumber: integer = 0; cTemp: real = 0);
      property SerialNumber: integer Read FSerialNumber;
      property Id:            string Read FId;
      property Residue:       string Read FResidue;
      property Subunit:         char Read FSubunit;
      property ResNumber:    integer Read FResNumber;
      property Temp:            real Read FTemp;
      function WriteAtomPDB: string;
  end;
```

Descendiente de la clase TAtom, de la que hereda todas sus propiedades y métodos. Permite representar la estructura de datos de átomos de un fichero PDB. Por ello añade seis nuevos campos privados con los atributos característicos de una línea ATOM de un fichero PDB (ver Figura 2). Dispone de los métodos siguientes:

constructor CreateAtomPDB(cx: real = 0; cy: real = 0; cz: real = 0; cmarked: boolean = false; ccolor: TColor = clwhite; cradium: real = 1; cSerialNumber: integer = 0; cId: string = 'NA'; cresidue: string = 'NA'; cSubunit: char = ' '; cResNumber: integer = 0; cTemp: real = 0).- Permite crear instancias de clase iniciadas a valores específicos de todos sus campos. Permite, además, crear instancias vacías (sin parámetros) o solo parcialmente iniciadas a valores específicos. Las instancias creadas con este constructor deben destruirse, cuando ya no se necesiten, con el destructor "Free".

[Ejemplo]:

```
var
  MiAtomo: TAtomPDB;
…
…
begin
  Miátomo:= TAtomPDB.CreateAtomPDB(1,2,3);
  …
  MiAtomo.Free;
End;
```

crea una instancia de TAtomPDB en la posición indicada, radio unidad, color blanco y marca "false", Número de Serie=0, identificador: "NA", residuo: "NA", subunidad: ' ', Número de residuo: 0 y Factor de temperatura: 0.

property SerialNumber: integer Read FSerialNumber.- Es una propiedad de solo lectura que permite obtener (pero no asignar) el valor del campo "Número de Serie" del átomo en el fichero PDB (FSERIALNUMBER).

property Id: string Read Fid.- Es una propiedad de solo lectura que permite obtener (pero no asignar) el valor del campo "identificador" (símbolo de uno o varios caracteres asignados a cada tipo de átomo en función del símbolo elemental y la posición dentro de cada residuo: FID).

property Residue: string Read FResidue.- Es una propiedad de solo lectura que permite obtener (pero no asignar) el valor del campo "Residuo" (residuo al que pertenece el átomo en el fichero PDB: FRESIDUE).

property Subunit: char Read FSubunit.- Es una propiedad de solo lectura que permite obtener (pero no asignar) el valor del campo "Subunidad" (subunidad a la que pertenece el átomo en el fichero PDB : FSUBUNIT).

property ResNumber: integer Read FResNumber.- Es una propiedad de solo lectura que permite obtener (pero no asignar) el valor del campo "Número de residuo" (número asignado en el fichero PDB al residuo al que pertenece el átomo: FRESNUMBER).

property Temp: real Read FTemp.- Es una propiedad de solo lectura que permite obtener (pero no asignar) el valor del campo "Factor de temperatura" (factor que pondera la movilidad del átomo y por tanto la calidad del dato cristalográfico en un fichero PDB : FTEMP).

function WriteAtomPDB: string.- Permite reconstruir una línea "ATOM" con la sintaxis correcta del protocolo PDB, a partir de sus datos (es decir, a partir de la instancia TAtomPDB del átomo). Devuelve una cadena de caracteres con la línea completa.

Clase TRESIDUOPDB

```
  TresiduePDB = Class  //-----------------------------------------------------
    private
      FResNumber: integer;
      FSubunit: char;
      FId3: string;
      FId1: char;
      FFirstPos, FLastPos: integer;
      FN, FCA, FC, FO: integer;
      Fphi, Fpsi: real;
      FPDB: ^TPDB;
    protected
      function NAtomPDB:  TAtomPDB;
    public
      constructor CreateResiduePDB(cResNumber: integer = 0; cSubunit: char = '
';
                  cId3: string = 'NA'; catm1: integer = 0; cAtmN: integer = 0;
                  cN: integer = 0; cCA: integer = 0; cC: integer = 0;
                  cO: integer = 0; cPhi: real = MinFloat; cPsi: real =
MinFloat);
      property ResNumber: integer read FResNumber;
      property Subunit: char read FSubunit;
      property  Id3:  string read FId3;
      property  Id1:  char read FId1;
      property FirstPos: integer read FFirstPos;
      property LastPos:  integer read FLastPos;
      property    NPos:  integer read FN;
      property   CAPos:  integer read FCA;
      property    CPos:  integer read FC;
      property    OPos:  integer read FO;
      property  phi:  real read Fphi;
      property  psi:  real read Fpsi;
      function CAAtomPDB: TAtomPDB;
      function CAtomPDB:  TAtomPDB;
      function OAtomPDB:  TAtomPDB;
      function FirstAtomPDB: TAtomPDB;
      function LastAtomPDB:  TAtomPDB;
  end;
```

Permite representar la estructura de datos de un residuo de un fichero PDB. Para ello dispone de trece campos privados accesibles a través de las correspondiente propiedades y funciones. Son los siguientes:

FPDB: es un puntero a la proteína a la que corresponde el residuo. Sirve, por ejemplo, para referenciar átomos de un PDB a partir de un residuo. Solo debe emplearse en el contexto de desarrollo de nuevos métodos de clase TPDB. Nunca fuera de ellos.

FPhi y FPsi: son valores reales de los ángulos de torsión “Fi” y “Psi” de ese residuo.

FN, FCA, FC y FO: son valores enteros que apuntan a las posiciones de esos átomos en el vector ATM de una estructura TPDB. Más adelante se volverá a ello. No confundir con los números de serie de estos átomos!

FFirstPos y FLastPos: son valores enteros que apuntan las posiciones del primero y último átomo del residuo en el vector ATM de una estructura TPDB. No confundir con los números de serie de estos átomos!

FId1 y FId3: (char y string, respectivamente) son los códigos de una y tres letras del aminoácido correspondiente a ese residuo.

FSubUnit: es un carácter único (char) que identifica a la subunidad a la que pertenece el residuo. Así por ejemplo, si un residuo pertenece a la tercera subunidad de una proteína, su FSubUnit suele ser o el carácter '3' o el carácter 'C'. Cuando solo existe una única subunidad, por conveniencia, se le asigna una cadena con un espacio: ' '.

FResNumber: Número de serie del residuo que figura en el fichero PDB. Debe tenerse en cuenta que el número de residuo es un número de orden que se inicia con cada una de las subunidades. Por ejemplo, para una proteína con cuatro subunidades como la hemoglobina hay 4 residuos cuyo número de serie es el 3. (Hay un residuo nº 3 para cada una de las subunidades).

La clase TResiduoPDB dispone de los métodos siguientes:

constructor CreateResiduePDB(cResNumber: integer = 0; cSubunit: char = ' '; cId3: string = 'NA'; catm1: integer = 0; cAtmN: integer = 0; cN: integer = 0; cCA: integer = 0; cC: integer = 0; cO: integer = 0; cPhi: real = MinFloat; cPsi: real = MinFloat).- Permite crear instancias de la clase con valores iniciados. Como en los constructors anteriormente descritos, también es posible crear instancias "vacías" o "semi-llenas".

property ResNumber: integer read FResNumber.- Es una propiedad de solo lectura que permite obtener (pero no asignar) el valor del campo "Número de residuo" (número asignado en el fichero PDB al residuo: FRESNUMBER).

property Subunit: char read FSubunit.- Es una propiedad de solo lectura que permite obtener (pero no asignar) el valor del campo "Subunidad" (subunidad a la que pertenece residuo en el fichero PDB : FSUBUNIT).

property Id3: string read FId3.- Es una propiedad de solo lectura que permite obtener (pero no asignar) el valor del campo "código de tres letras" (terna de caracteres asignados a cada tipo de residuo: FID3).

property Id1: char read FId1.- Es una propiedad de solo lectura que permite obtener (pero no asignar) el valor del campo "código de una letra" (carácter asignado en la nomeclatura moderna a cada tipo de residuo: FID1).

property FirstPos: integer read FFirstPos.- Propiedad de solo lectura que permite obtener (pero no asignar), como valor entero, la posición del primer átomo del residuo en el vector ATM de una estructura TPDB. No confundir con los números de serie de estos átomos!

property LastPos: integer read FLastPos.- Propiedad de solo lectura que permite obtener (pero no asignar), como valor entero, la posición del último átomo del residuo en el vector ATM de una estructura TPDB. No confundir con los números de serie de estos átomos!

property NPos: integer read FN.- Propiedad de solo lectura que permite obtener (pero no asignar), como valor entero, la posición del átomo de nitrógeno alfa-amino del residuo en el vector ATM de una estructura TPDB. No confundir con los números de serie de estos átomos!

property CAPos: integer read FCA.- Propiedad de solo lectura que permite obtener (pero no asignar), como valor entero, la posición del carbono alfa del residuo en el vector ATM de una estructura TPDB. No confundir con los números de serie de estos átomos!

property CPos: integer read FC.- Propiedad de solo lectura que permite obtener (pero no asignar), como valor entero, la posición del carbono alfa-carbonílico del residuo en el vector ATM de una estructura TPDB. No confundir con los números de serie de estos átomos!

property OPos: integer read FO.- Propiedad de solo lectura que permite obtener (pero no asignar), como valor entero, la posición del oxígeno alfa-carbonílico del residuo en el vector ATM de una estructura TPDB. No confundir con los números de serie de estos átomos!

property phi: real read Fphi.- Propiedad de solo lectura que permite obtener (pero no asignar), como valor real, el valor en radianes del ángulo de torsión "Fi" del residuo actual "r", determinado por los átomos C_{r-1}, N_r, Ca_r y C_r.

property psi: real read Fpsi.- Propiedad de solo lectura que permite obtener (pero no asignar), como valor real, el valor en radianes del ángulo de torsión "Psi" del residuo actual "r", determinado por los átomos N_r, Ca_r y C_r y N_{r+1}.

function CAAtomPDB: TAtomPDB.- Función que devuelve un TAtomPDB con los datos del carbono alfa del residuo. Solo debe emplearse en el contexto de desarrollo de nuevos métodos de clase TPDB. Nunca fuera de ellos.

function CAtomPDB: TAtomPDB.- Función que devuelve un TAtomPDB con los datos del carbono alfa-carbonílico del residuo. Solo debe emplearse en el contexto de desarrollo de nuevos métodos de clase TPDB. Nunca fuera de ellos.

function OAtomPDB: TAtomPDB.- Función que devuelve un TAtomPDB con los datos del oxígeno alfa-carbonílico del residuo. Solo debe emplearse en el contexto de desarrollo de nuevos métodos de clase TPDB. Nunca fuera de ellos.

function FirstAtomPDB: TAtomPDB.- Función que devuelve un TAtomPDB con los datos del primer átomo del residuo. Solo debe emplearse en el contexto de desarrollo de nuevos métodos de clase TPDB. Nunca fuera de ellos.

function LastAtomPDB: TAtomPDB.- Función que devuelve un TAtomPDB con los datos del último átomo del residuo. Solo debe emplearse en el contexto de desarrollo de nuevos métodos de clase TPDB. Nunca fuera de ellos.

Clase TSUBUNIDADPDB

```
TSubunitPDB = class //-----------------------------------------------------
  private
    FId: char;
    FFirstAtmPos, FLastAtmPos, FFirstResPos, FLastResPos: integer;
    FAtomCount, FResCount: integer;
    FResIndex: array of integer;
    Function GetResIndex(const i: integer): integer;
  public
    constructor CreateSubunitPDB(cResCount: integer = 0; cID: char = ' ';
                cAtm1: integer = 0; cAtmN: integer = 0; cRes1: integer = 0;
                cResN: integer = 0; cAtomCount: integer = 0);
    property Id:           char read FId;
    property FirstAtmPos: integer read FFirstAtmPos;
    property LastAtmPos:  integer read FLastAtmPos;
    property FirstResPos: integer read FFirstResPos;
    property LastResPos:  integer read FLastResPos;
    property AtomCount:   integer read FAtomCount;
    property ResCount:    integer read FResCount;
    property ResIndex[i: integer]: integer read GetResIndex;
    function ResSerialToRaw(const ResSerialNo: integer): integer;
    function MaxSerialResidue: integer;
end;
```

Permite representar la estructura de datos de un residuo de un fichero PDB. Para ello dispone de ocho campos privados accesibles a través de las correspondientes propiedades y funciones. Son los siguientes:

FId: es un carácter único (char) que identifica a la subunidad.

FFirstAtmPos: valor entero que apunta a la posición del primer átomo de la subunidad en el vector ATM de una estructura TPDB. No confundir con el número de serie de este átomo!

FLastAtmPos: valor entero que apunta a la posición del último átomo de la subunidad en el vector ATM de una estructura TPDB. No confundir con el número de serie de este átomo!

FFirstResPos: valor entero que apunta a la posición del primer residuo de la subunidad en el vector RES de una estructura TPDB. No confundir con el número de serie de este residuo!

FLastResPos: valor entero que apunta a la posición del último residuo de la subunidad en el vector RES de una estructura TPDB. No confundir con el número de serie de este residuo!

FAtomCount: valor entero que contiene el número total de átomos de la subunidad.

FResCount: valor entero que contiene el número total de residuos de la subunidad.

FResIndex: vector de enteros que permite la indexación de residuos a partir de los subíndices del vector RES de un TPDB. (Más información en la descripción de la clase TPDB). Los subíndices de este vector corresponden a los números de serie (“residue sequence numbers”, ver Figura 2) de los residuos, tal y como aparecen en los ficheros PDB. Los contenidos de cada

elemento del vector corresponden a la numeración TPDB; esto es, a los subíndices de esos residuos en el vector RES de un TPDB. (Más información en la descripción de la clase TPDB).

La clase TSubUnidadPDB dispone de los métodos siguientes:

constructor CreateSubunitPDB(cResCount: integer = 0; cID: char = ' '; cAtm1: integer = 0; cAtmN: integer = 0; cRes1: integer = 0; cResN: integer = 0; cAtomCount: integer = 0).- Permite crear instancias de la clase TSubUnidadPDB. Permite, por lo tanto representar las estructuras de datos de las subunidades de una proteína.

property Id: char read Fid.- Es una propiedad de solo lectura que permite obtener (pero no asignar) el valor del campo "identificador" de una subunidad (carácter único asignado a cada subunidad: FID).

property FirstAtmPos: integer read FFirstAtmPos.- Propiedad de solo lectura que permite obtener (pero no asignar), como valor entero, la posición del primer átomo de la subunidad en el vector ATM de una estructura TPDB. No confundir con su números de serie!

property LastAtmPos: integer read FLastAtmPos.- Propiedad de solo lectura que permite obtener (pero no asignar), como valor entero, la posición del último átomo de la subunidad en el vector ATM de una estructura TPDB. No confundir con su números de serie!

property FirstResPos: integer read FFirstResPos.- Propiedad de solo lectura que permite obtener (pero no asignar), como valor entero, la posición del primer residuo de la subunidad en el vector RES de una estructura TPDB. No confundir con su número de serie ("residue sequence numbers")!

property LastResPos: integer read FLastResPos.- Propiedad de solo lectura que permite obtener (pero no asignar), como valor entero, la posición del último residuo de la subunidad en el vector RES de una estructura TPDB. No confundir con su número de serie ("residue sequence numbers")!

property AtomCount: integer read FAtomCount.- Propiedad de solo lectura que permite obtener (pero no asignar), como valor entero, el número total de átomos de la subunidad.

property ResCount: integer read FResCount.- Propiedad de solo lectura que permite obtener (pero no asignar), como valor entero, el número total de residuos de la subunidad.

property ResIndex[i: integer]: integer read GetResIndex.- Propiedad de solo lectura que permite obtener (pero no asignar), como valor entero, el subíndice del elemento del vector RES de una estructura TPDB que contiene los datos de ese residuo (en lo sucesivo, numeración "propia" o numeración "raw").

function ResSerialToRaw(const ResSerialNo: integer): integer.- Función entera, equivalente a la anterior, que devuelve la numeración "raw" de un residuo en el vector RES de una estructura TPDB a partir de su número de serie ("residue sequence number"). Dicho de

otro modo, devuelve el subíndice del elemento del vector RES de un TPDB que contiene los datos de ese residuo.

function MaxSerialResidue: integer.- Función entera que devuelve el mayor valor de número de serie de un residuo ("residue sequence number") en la subunidad.

Clase TPDB

```
TPDB = class  //-----------------------------------------------------------------
    private
      Fheader: string;
      Fatm: array of TAtomPDB;
      Fres: array of TResiduePDB;
      Fsub: array of TSubUnitPDB;
      FAtomCount, FResCount, FSubCount: integer;
      Fsequence, Fsubunits: string;
      FAtmIndex: array of integer;
      FmaxSerial: integer;
      Function GetAtm      (const i: integer): TAtomPDB;
      Function GetRes      (const i: integer): TResiduePDB;
      Function GetSub      (const i: integer): TSubUnitPDB;
      Function GetAtmIndex (const i: integer): integer;
    public
      constructor CreatePDB;
      Destructor destroy;  override;
      property atm[i: integer]: TAtomPDB      read GetAtm; default;
      property res[i: integer]: TResiduePDB   read GetRes;
      property sub[i: integer]: TSubUnitPDB read GetSub;
      property header:    string read Fheader write Fheader;
      property AtomCount: integer read FAtomCount;
      property ResCount: integer read FResCount;
      property SubCount: integer read FSubCount;
      property Sequence:  string read FSequence;
      property Subunits:  string read FSubUnits;
      property AtmIndex[i: integer]: integer read GetAtmIndex;
      property MaxSerial: integer read FMaxSerial;
      function SerialToRaw(const SerialNo: integer): integer;
      function ResSerialToRaw(const ResSerialNo: integer;
                              const SubNo: integer): integer;
      function NfromRes(const resno: integer): TAtomPDB;
      function CAfromRes(const resno: integer): TAtomPDB;
      function CfromRes(const resno: integer): TAtomPDB;
      function OfromRes(const resno: integer): TAtomPDB;
      function resizePDB(const newlength: integer=0): boolean;
      function LoadPDB(Sender: TObject; PDBtext: TStrings): boolean;
      function LoadPDB: string;
      function MaxSerialNumber: integer;
      function AtomtoARS(AtomNo: integer): TARS;
      function SerialtoARS(AtomNo: integer): TARS;
      function ResnotoARS(ResSerial: integer; SubChar: Char): TARS;
      procedure resetPDB;
      Procedure InitPanel(AtomSpin, ResSpin: TSpinEdit;
                    SubCombo: TComboBox; Track: TTrackBar; mode: integer = 0);
      Procedure RefreshPanelFromAtom(AtomSpin, ResSpin: TSpinEdit;
                              SubCombo: TComboBox; Track: TTrackBar);
      Procedure RefreshPanelFromResidue(AtomSpin, ResSpin: TSpinEdit;
                              SubCombo: TComboBox; Track: TTrackBar);
      Class Function AA3to1(s: string): char;
  end;
```

Es la clase que contiene todos los datos de una proteína extraídos a partir de un fichero PDB. Se trata por lo tanto de la estructura de datos correspondiente a una proteína completa. La clase contiene tres vectores privados FAtm, Fres y FSub que contienen, respectivamente, los datos de los átomos, residuos y subunidades de la proteína (sus elementos son, por tanto, de tipo TAtomPDB, TResiduePDB y TSubUnitPDB). Se tratan de matrices abiertas que deben alojarse dinámicamente en memoria mediante la función Setlength() una vez se conoce el número de elementos de la proteína y que empiezan por el elemento 1. (El elemento 0 es ignorado, por lo tanto). El orden de átomos, residuos y subunidades es un orden correlativo (que llamaremos numeración "propia" o "raw") y que, en principio no tiene por qué coincidir con la numeración asignada en los ficheros PDB a cada átomo ("atom serial number") o a cada residuo ("residue sequence number"). Ver figura 2.

Junto a estas estructuras, TPDB contiene siete campos privados y un vector adicional de indexación de átomos, también privado, que facilita la conversión entre los dos sistemas de numeración mencionados. El acceso a estos miembros privados se hace con una serie de propiedades, funciones y procedimientos públicos que se describirán a continuación. Entre ellos, destacan las dos funciones sobrecargadas LoadPDB(), que sirven para incorporar todos los datos de la proteína a partir del fichero de texto PDB, como también se verá más adelante.

Los miembros privados son los siguientes:

Fheader[]: Cadena correspondiente al la primera línea del fichewro PDB. Contiene la descripción sucinta de la proteína.

FAtm[]: Vector abierto de átomos (TAtomPDB). Contiene toda la información de las líneas ATOM del fichero PDB y su primer subíndice es el 1.

Fres[]: Vector abierto de residuos (TResiduePDB). Contiene el registro de todos los residuos de la proteína, numerados consecutivamente desde el subíndice 1.

FSub[]: Vector abierto de subunidades ()TSubUnitPDB). Contiene los registros de todas las subunidades de la proteína, numerados consecutivamente desde el subíndice 1.

FAtmIndex[]: Vector de indexación de átomos a partir de su numeración asignada en el fichero de texto PDB. Permite el acceso directo a la numeración "raw" correspondiente a los subíndices del vector FATM[].

FAtomCount: valor entero que se corresponde con el número total de átomos de la proteína

FResCount: valor entero que se corresponde con el número total de residuos de la proteína

FSubCount: valor entero que se corresponde con el número total de subunidades de la proteína

FSequence: cadena de caracteres que se corresponde con la secuencia total de la proteína en códigos de una letra de sus residuos correspondientes. La cadena abarca todas las subunidades en el orden de aparición.

FSubUnits: Cadena de caracteres que se corresponde con la secuencia de identificadores (de un solo carácter) de las subunidades que componen la proteína.

FMaxSerial: Máximo “atom serial number” de la proteína. Se requiere porque la numeración PDB puede no ser consecutiva o estar incompleta. Es por ello que empleamos la numeración propia o “raw”, para evitar posibles inconsistencias.

La clase TPDB dispone de los métodos siguientes:

constructor CreatePDB.- Permite crear instancias “vacías” de la clase TPDB. Completamente equivalente al constructor por defecto CREATE.

Destructor destroy; override.- Destructor de las instancias de TPDB. Debe invocarse para destruir una instancia, cuando ya no se necesita. Preferible al “Free”.

property atm[i: integer]: TAtomPDB read GetAtm; default.- Propiedad de solo lectura que permite obtener (pero no asignar) un TAtomPDB con los datos del átomo indicado por su numeración Raw en el vector FAtm. Es la propiedad por defecto. Ello quiere decir que se puede abreviar: [Ejemplo, para un TPDB denominado “p”]: p.atm[i] es equivalente a p[i]

property res[i: integer]: TResiduePDB read GetRes.- Propiedad de solo lectura que permite obtener (pero no asignar) un TResiduePDB con los datos del residuo indicado por su numeración Raw en el vector FAtm. property sub[i: integer]: TSubUnitPDB read GetSub;

property header: string read Fheader write Fheader.- Propiedad de lectura y escritura que permite obtener o asignar una cadena con el contenido de la primera línea del fichero de texto PDB (información general de la proteína).

property AtomCount: integer read FAtomCount.- Propiedad de solo lectura que permite obtener (pero no asignar) el número total de átomos de la proteína.

property ResCount: integer read FResCount.- Propiedad de solo lectura que permite obtener (pero no asignar) el número total de residuos de la proteína.

property SubCount: integer read FSubCount.- Propiedad de solo lectura que permite obtener (pero no asignar) el número total de subunidades de la proteína.

property Sequence: string read FSequence.- Propiedad de solo lectura que permite obtener, como cadena, la secuencia completa de la proteína como código de una letra de sus residuos.

property Subunits: string read FSubUnits.- Propiedad de solo lectura que permite obtener, como cadena, la secuencia completa de la proteína como código de una letra de sus subunidades.

property AtmIndex[i: integer]: integer read GetAtmIndex.- Propiedad de solo lectura que permite obtener, como valor entero, la numeración “raw” de un átomo a partir de su “serial number”.

property MaxSerial: integer read FMaxSerial.- Propiedad de solo lectura que permite obtener, como valor entero, el máximo atom serial number de la proteína.

function SerialToRaw(const SerialNo: integer): integer.- Función equivalente a la propiedad AtmIndex[] que permite obtener, como valor entero, la numeración "raw" de un átomo a partir de su "serial number".

function ResSerialToRaw(const ResSerialNo: integer; const SubNo: integer): integer.- Función entera que permite obtener, la numeración "raw" (subíndice del vector RES) de un residuo a partir de su "residue sequence number".

function NfromRes(const resno: integer): TAtomPDB.- Función que devuelve el TAtomPDB del nitrógeno alfa-amino de un residuo a partir de su numeración "raw".

function CAfromRes(const resno: integer): TAtomPDB.- Función que devuelve el TAtomPDB del carbono alfa de un residuo a partir de su numeración "raw".

function CfromRes(const resno: integer): TAtomPDB.- Función que devuelve el TAtomPDB del carbono alfa-carbonílico de un residuo a partir de su numeración "raw".

function OfromRes(const resno: integer): TAtomPDB.- Función que devuelve el TAtomPDB del oxígeno alfa-carbonílico de un residuo a partir de su numeración "raw".

function resizePDB(const newlength: integer=0): Boolean.- Función que permite redimensional las matriz Atom de un PDB.

function LoadPDB(Sender: TObject; PDBtext: TStrings): Boolean.- Función que permite cargar los datos estructurales de una proteína a partir de un fichero PDB en una estructura TPDB. Como parámetros toma el objeto que la invoca y el texto (como TStrings) de un fichero PDB. Es la función que ejecuta el "parsing" completo de los datos y, en general, es llamada por la otra función sobrecargada con el mismo nombre (a continuación). También puede ser invocada de forma directa, pero en tal caso la carga del fichero PDB debe hacerse de forma previa y externa. Devuelve un valor "boolean" que informa si el resultado es o no fallido.

function LoadPDB: string.- Es la función de carga que se debe invocar por regla general. No requiere la gestión previa de la carga del fichero ni su volcado en un objeto de tipo TMemo o de tipo TStrings, como ocurría con la función anterior. Devuelva un "string" con la ruta completa y el nombre de la proteína cargada.

function MaxSerialNumber: integer.- Función equivalente a la propiedad MAXSERIAL. Devuelve, como valor entero, el máximo "atom serial number" de la proteína.

function AtomtoARS(AtomNo: integer): TRAS.- Convierte a formato TARS (tipo "Atom-Residue-SubUnit") los datos de un átomo, a partir de su "Raw number", que toma como argumento.

function SerialtoARS(AtomNo: integer): TARS.- Convierte a formato TARS (tipo "Atom-Residue-SubUnit") los datos de un átomo, a partir de su "Atom serial number", que toma como argumento.

function ResnotoARS(ResSerial: integer; SubChar: Char): TARS.- Convierte a format TARS (tipo "Atom-Residue-SubUnit") los datos del último átomo de un residuo, a partir de su "Residue sequence number", que toma como argumento.

procedure resetPDB.- Procedimiento que "vacía" el contenido de una instancia de tipo TPDB, eliminando todos los datos de la proteína cargada. Permite reutilizar la instancia para cargar una nueva proteína.

Procedure InitPanel(AtomSpin, ResSpin: TSpinEdit; SubCombo: TComboBox; Track: TTrackBar; mode: integer = 0).- Un panel se define como el conjunto de los cuatro componentes visuales (controles) siguientes: dos objetos TSpinEdit que indican el número de átomo y el número de residuo; un objeto de tipo TComboBox que indica la subunidad y un objeto de tipo TrackBar que conmuta entre numeración "raw" y "atom serial number". El panel se construye para responder de forma coordinada a la modificación de uno de sus valores para dar siempre una combinación válida de Átomo-Residuo-Subunidad. Es útil para crear una interfaz interactiva donde deban seleccionarse átomos o residuos de una proteína para su posterior procesamiento. La función InitPanel, inicia este conjunto de controles y lo prepara para una respuesta coordinada a cualquier modificación del contenido de uno de ellos. (Ver aplicación de demostración)

Procedure RefreshPanelFromAtom(AtomSpin, ResSpin: TSpinEdit; SubCombo: TComboBox; Track: TTrackBar).- Produce la respuesta combinada a la modificación del número de átomo de un panel previamente iniciado con INITPANEL. La introducción de un nuevo número de átomo provoca la modificación automática del control de residuo y de subunidad. El tipo de numeración ("raw" o "serial") se determina por el valor del objeto TRACKBAR del panel.

Procedure RefreshPanelFromResidue(AtomSpin, ResSpin: TSpinEdit; SubCombo: TComboBox; Track: TTrackBar).- Produce la respuesta combinada a la modificación del número de residuo (el "residue sequence number") de un panel previamente iniciado con INITPANEL. La introducción del nuevo número de residuo provoca la modificación automática del control de residuo y de subunidad. La ejecución de este procedimiento determina que el control de visualización de número del átomo se modifique al último átomo del residuo seleccionado.

Class Function AA3to1(s: string): char.- Convierte la nomenclatura de tres dígitos de un aminoácido a la nomenclatura de un dígito. Como parámetro se le pasa un "string" con el código de tres letras. El resultado es un "Char" con el código de una letra del aminoácido en cuestión.

Clase V3

```
  V3 = Class  //-------------------------------------------------------------
-
    public
      Class Function resize      (factor: real; vector: TTransform):
TTransform;
      Class Function resize      (vector: TTransform; factor: real):
TTransform;
      Class Function Module      (vector:     TTransform): real;    // módulo
      Class Function Distance    (v1, v2:     TTransform): real;
      Class Function Sum         (v1, v2:     TTransform): TTransform;
      Class Function Diff        (v1, v2:     TTransform): TTransform;
      Class Function DotProduct  (v1, v2:     TTransform): real;
      Class Function CrossProduct(A, B:       TTransform): TTransform;
      Class Function Angle       (v1, v2:     TTransform): real;
      Class Function VertexAngle (A, B, C:    TTransform): real;   // ángulo
ABC
      Class Function Torsion     (A, B, C, D: TTransform): real;
  end;
```

La clase V3 agrupa exclusivamente funciones de clase que pueden ser invocadas externamente sin necesidad de instanciar objetos de tipo V3. En este momento dispone de las siguientes funciones:

Class Function resize (factor: real; vector: TTransform): TTransform.- Función de escalamiento con dos versiones sobrecargadas que se diferencian exclusivamente en el orden de los argumentos. Devuelve el vector pasado como argumento escalado por un factor único que afecta a las tres dimensiones, de forma isométrica.

Class Function resize (vector: TTransform; factor: real): TTransform.- Función de escalamiento con dos versiones sobrecargadas que se diferencian exclusivamente en el orden de los argumentos. Devuelve el vector pasado como argumento escalado por un factor único que afecta a las tres dimensiones, de forma isométrica.

Class Function Module (vector: TTransform): real.- Devuelve el modulo de un vector como valor real;

Class Function Distance (v1, v2: TTransform): real.- Devuelve la distancia Euclidea entre los dos vectores que se pasan como argumentos.

Class Function Sum (v1, v2: TTransform): TTransform.- Devuelve el vector suma de los vectores pasados como argumentos.

Class Function Diff (v1, v2:TTransform): TTransform.- Devuelve el vector diferencia de los vectores pasados como argumentos, de los que el primero sería el minuendo y el segundo el substraendo.

Class Function DotProduct (v1, v2: TTransform): real.- Devuelve un valor real igual al product scalar de los vectores pasados como argumentos.

Class Function CrossProduct(A, B: TTransform): TTransform.- Devuelve un vector que es el producto vectorial de los dos vectores pasados como argumentos.

Class Function Angle (v1, v2: TTransform): real.- Devuelve el ángulo en radianes formado por los dos vectores pasados como argumentos.

Class Function VertexAngle (A, B, C: TTransform): real.- Devuelve el ángulo en radianes determinado por tres vectores de posición de los que el segundo de los argumentos sería el vértice.

Class Function Torsion (A, B, C, D: TTransform): real.- Devuelve el ángulo de torsion, en radianes, determinado por los cuatro vectores de posición pasados como argumento, de los que el segundo y el tercero, determinan el eje de torsión. El resultado que produce esta función está corregido según los criterios establecidos por la IUPAC-IUB y, por lo tanto, no coinciden necesariamente con el ángulo diedro determinado por los vectores de posición A, B, C y D.

Sobrecarga de Operadores

La librería incluye también una serie de funciones basadas en operadores sobrecargados. Son las siguientes:

Operator + (v1, v2: TTransform) z: TTransform.- Es el operador suma de vectores. Sus operandos son dos vectores de tipo TTransform y su resultado es otro vector del mismo tipo.

Operator - (v1, v2: TTransform) z: TTransform.- Es el operador diferencia de vectores. Sus operandos son dos vectores de tipo TTransform. El primero actúa de minuendo y el segundo de sustrendo. El resultado es otro TTransform.

Operator - (vector: TTransform) z: TTransform.- Operador ‘opuesto de un vector’. Devuelve un vector con igual módulo y dirección pero con sentido contrario al operando.

Operator * (factor: real; vector: TTransform) z: TTransform;// escalar*vector.- Operador ‘Producto de un escalar por un vector’. Devuelve el vector escalado por el factor. Hay dos versiones sobrecargadas de este operador que se diferencian exclusivamente en el orden de los operandos.

Operator * (vector: TTransform; factor: real) z: TTransform;// vector*escalar.- Operador ‘Producto de un escalar por un vector’. Devuelve el vector escalado por el factor. Hay dos

versiones sobrecargadas de este operador que se diferencian exclusivamente en el orden de los operandos.

Operator * (v1, v2: TTransform) z: real.- Operador ‘producto escalar’. Devuelve un real que equivale al producto escalar de los vectores operandos.

Operator ** (v1, v2: TTransform) z: TTransform.- Operador ‘producto vectorial’. Devuelve el vector producto vectorial de los vectores operandos.

Operator < (v1, v2: TTransform) z: real.- Ángulo en radianes formado por los dos vectores que se pasan como operandos.

Operator / (v1, v2: TTransform) z: real.- Operador ‘Distancia Euclidea’. Calcula la distancia euclidea entre los vectores operandos.