ESTRUCTURA DE DATOS PLEX

Ardenghi, Jorge
Cicileo, Guillermo
Rueda, Sonia
Zanconi, Marcelo

CRIBABB-PLAPIQUI 12 de Octubre 1865
(8000) Bahía Blanca República Argentina

Resumen

La mayor parte de los problemas encierran en si mismos una estructura. Así, un problema puede ser especificado en término de los objetos que lo caracterizan, sus relaciones y las operaciones que se ejecutan sobre ellos.

Una característica deseable de un programa, es que respete, en la medida de lo posible, la estructura del problema que intenta resolver.

La estructura de datos Plex permite la representación de los datos de un problema consistente de entidades, agrupadas en categorías y caracterizadas por atributos que reciben valores dentro de un dominio.

La aplicación de la estructura Plex provee gran flexibilidad, ya que, aunque las categorías y los dominios son fijos para un problema dado, es posible agregar nuevas entidades con una estructura propia.

Por otra parte la introducción de Tipos de Datos Abstractos permite el oscurecimiento de información a través del encapsulamiento de datos.

Estructura de Datos Plex

1.0 Estructuras de Datos

Una estructura de datos es una colección de items de datos organizados de alguna manera, junto con las operaciones para acceder a estos datos.

Una estructura de datos puede definirse en varios niveles :

- A **nivel abstracto o lógico** la estructura se define independientemente de una representación particular, sin connotaciones físicas. Las operaciones se definen desde el punto de vista de **que** es lo que hacen y no de **como** lo hacen.

- A **nivel operativo o de aplicación** la estructura se define en términos del tipos de problemas a los cuales se adapta. Es la parte más cercana al usuario de la estructura. En ocasiones se escribe una interfase, de forma tal que el usuario tiene acceso a las operaciones sobre la estructura a través de operaciones más cercanas a su aplicación particular.

- A **nivel físico o de implementación** la estructura se define en términos de otras estructuras más simples, creadas posiblemente a partir de mecanismos y facilidades provistos por el lenguaje. Esta descripción puede dividirse a su vez en subniveles, de forma tal que los niveles superiores son aún algo abstractos, mientras que los inferiores son mucho más cercanos a la implementación particular y a las primitivas provistas por el lenguaje. Las operaciones se definen como procedimientos que manipulan a la estructura en términos de operaciones primitivas.

1.1 Definicion De Plex A Nivel Logico

Informalmente se puede pensar a la estructura Plex como un conglomerado de **celdas**, conjunto de datos con cohesión conceptual, agrupadas de alguna manera. Las celdas varían en número y tipo de los datos constituyentes llamados **componentes**. Cada celda puede estar asociada con cero o más celdas de la misma u otra agrupación.

Más formalmente se dice que a nivel lógico la estructura de datos Plex se define como una estructura **dinámica, heterogénea, no lineal** y **multiasociada**.

Dinámica: porque puede crecer y contraerse tanto en el número de elementos como en sus tamaños.

Heterogenea: porque los elementos son potencialmente distintos en número y

tipo de componentes y pueden estar agrupados en subestructuras.

No lineal: porque no existe una relación de orden entre sus elementos.

Multiasociada: porque cada elemento puede estar asociado a cero o más elementos.

Se da ahora una definición más **operativa** de la estructura, apuntando al tipo de problemas a que se adapta.

1.2 Definicion De Plex A Nivel Aplicacion

La mayor parte de los problemas encierran en si mismos una estructura.

Es deseable, aunque no siempre posible, que cuando los datos referidos a un problema se almacenan en una computadora, éstos esten organizados respetando la estructura inherente al problema.

Una estructura de datos plex permite almacenar la descripción de un problema caracterizado por:

1) Objetos distinguibles que llamaremos **entidades**.

2) Las entidades pueden estar agrupadas en **categorías** cuando representan a objetos de un mismo tipo.

3) Una entidad está caracterizada por ciertas propiedades denominadas **atributos**.

4) Cada atributo toma valores dentro de un **dominio**.

5) Un atributo común a todas las entidades es el **nombre** que permite identificarla.

6) Dos o más entidades pueden estar asociadas a un mismo conjunto de atributos, en especial si pertenecen a una misma categoría, pero en general tendrán distintos valores para uno o más atributos.

7) En general no existen dos entidades de distintas categorías asociadas un mismo conjunto de atributos.

8) Las entidades pueden estar **asociadas** a entidades de la misma y/o de otras categorías.

1.3 Definicion De Plex A Nivel Implementacion

La implementación de una estructura de datos compleja puede presentarse en distintos niveles de abstracción. En los niveles más altos se describe a la estructura en términos de otras estructuras más simples. Sin embargo si estas estructuras no son soportadas por el lenguaje de implementación, es posible que sea necesario definirlas a su vez en términos de su representación física, y asi siguiendo hasta alcanzar primitivas del lenguaje.

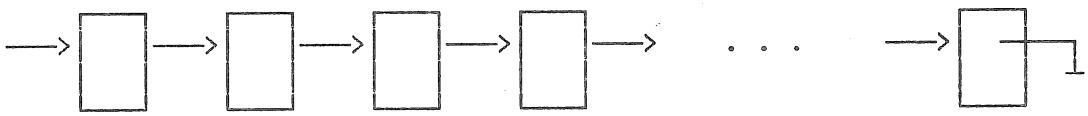
Se verá entonces una implementación posible para la estructura de datos plex, utilizando estructuras de datos más simples como listas y tablas hash, y se describirá luego como estas estructuras pueden representarse en Pascal.

1.3.1 Implementación en términos de otras estructuras

En una estructura de datos plex la información asociada a una entidad es almacenada en una **celda**. La celda es entonces, el elemento primitivo a partir del cual se construye la estructura.

Las celdas asociadas a entidades de una misma categoría están organizadas dentro de la estructura como una **lista enlazada**.

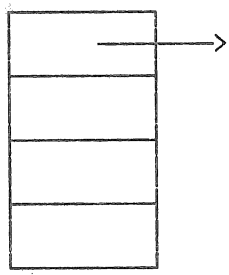
Categoria



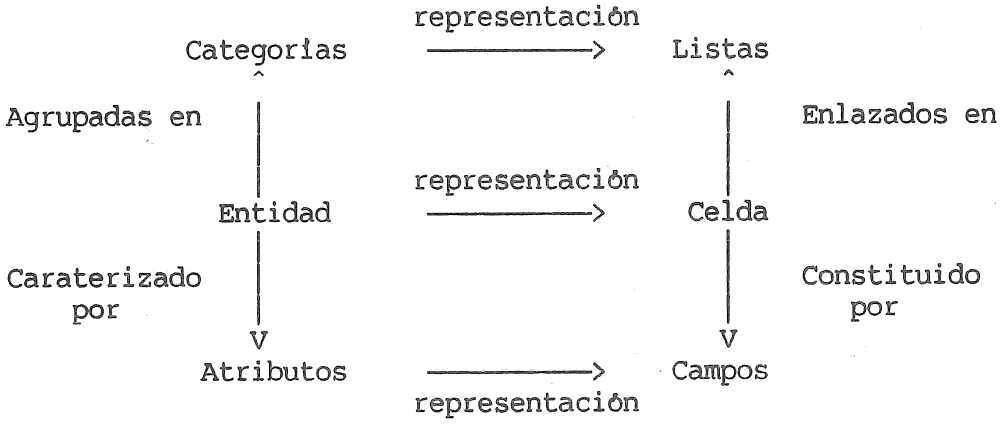
Entidad 1 Entidad 2 Entidad 3 Entidad 4 Entidad n

Una celda esta constituida por campos, cada uno de los cuales corresponde a un atributo de la entidad asociada a la celda, y por lo tanto recibe valores dentro de un dominio.

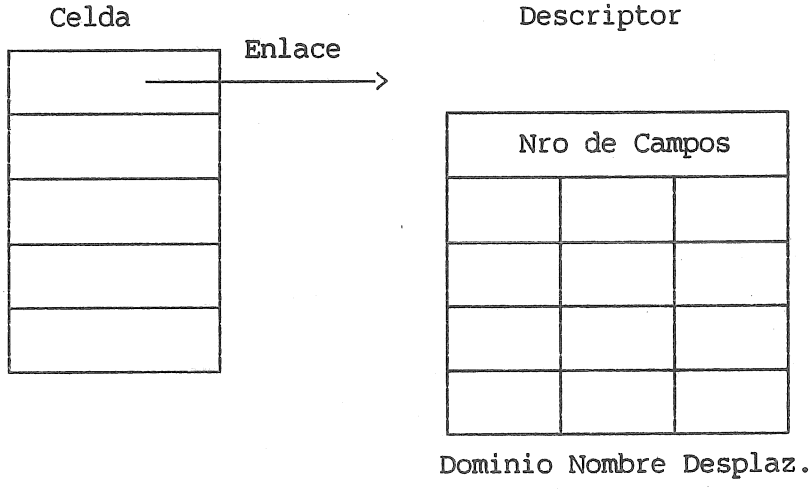
Celda



Gráficamente:

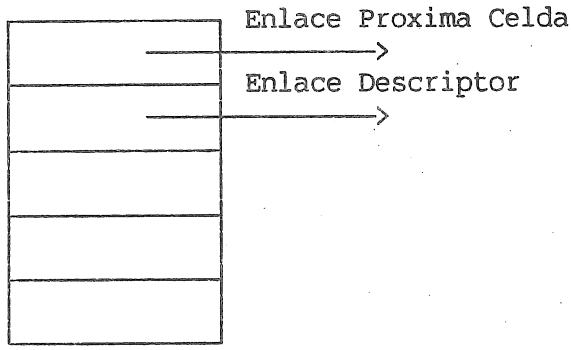


La estructura de una celda está definida a través de un descriptor asociado a la celda. Este permite almacenar el número de campos de la celda, y el dominio, nombre y desplazamiento de cada uno de ellos :



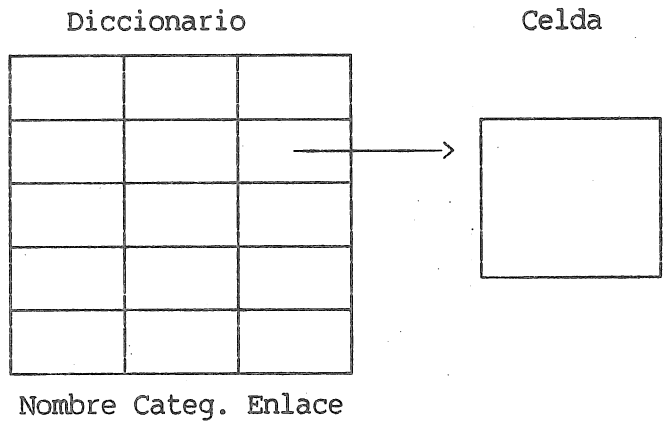
Por estar asociado a una entidad, cada celda tiene un nombre. Una celda tiene asignada además una dirección.

El número de campos de una celda es variable, pero contiene al menos dos campos fijos: la dirección de la próxima celda de la lista - si existe - y la dirección del descriptor. Dado que todas las celdas tienen estos dos campos, el descriptor no los incluye.



Una celda puede ser accedida por su nombre o por su dirección. Cuando se dispone del nombre, el acceso se realiza por medio de una estructura auxiliar : el diccionario.

El diccionario contiene una entrada de tres componentes por cada celda. La primer componente contiene el nombre de la celda y la segunda su dirección. Como cada descriptor también tiene asociado un nombre, el diccionario contiene además una entrada nombre-dirección por cada descriptor. Por cada entrada se almacena entonces un tercer valor, el tipo de entrada, que permite distinguir si el par nombre-dirección corresponde a un descriptor o a una celda.



Una lista enlazada de celdas se distingue entonces, de una lista enlazada común por dos hechos fundamentales. Primero no todas las componentes tienen necesariamente la misma estructura, segundo el acceso a una celda puede hacerse por medio del diccionario y no requiere recorrer la lista desde el principio. Estas dos características provocan que cada celda deba contener un puntero a su descriptor, porque aún en el caso de que todos los elementos de la lista tuvieran la misma estructura, al acceder a elementos interiores de la lista podemos no conocer por anticipado de que tipo de celda se trata.

Aunque la representación completa depende de la aplicación particular, estamos en condiciones de introducir ciertas características de la estructura general.

La estructura se completa con una cabeza conteniendo información general, como por ejemplo, punteros a las listas de celdas, punteros a las primeras locaciones libres en el pool, en la tabla de descriptores, etc.

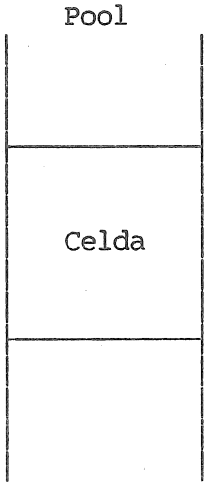
Tanto la cabeza como el diccionario y los descriptores, contienen unicamente información estructural para permitir almacenar y consultar la información relativa al proceso.

La información propiamente dicha reside en las celdas. Una celda contiene siempre parte de la información que describe al problema. Sin embargo también contienen parte de la información estructural. Esto es, las celdas contienen un puntero a un descriptor y al menos un puntero a otra celda - la próxima de la lista -.

1.3.2 Implementación en términos de primitivas de un Lenguaje

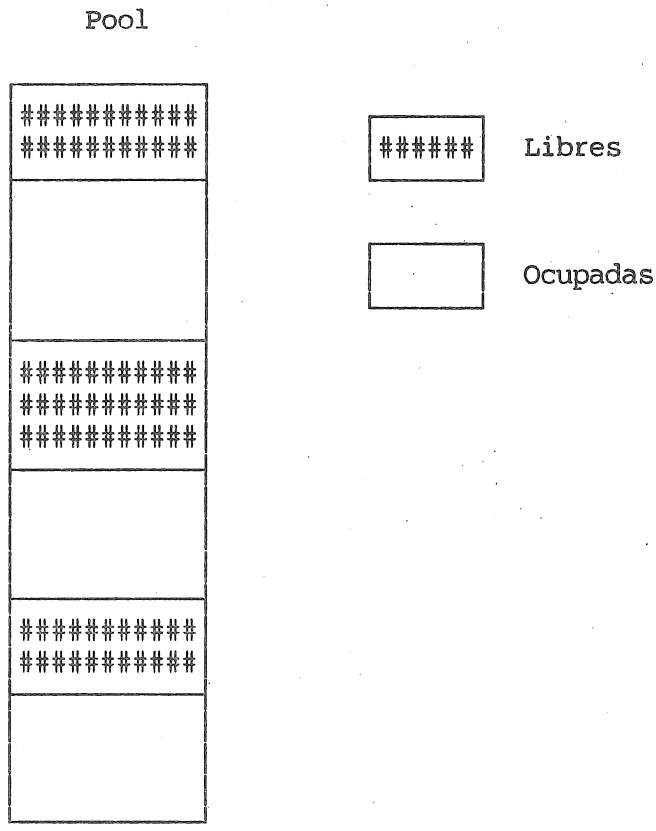
Las operaciones primitivas para manipulación de la estructura plex están implementadas en Pascal. La decisión para adoptar este lenguaje se basó fundamentalmente en la disponibilidad de constructores de tipos de datos muy flexibles y en la posibilidad de combinarlos, formando estructuras más complejas. Por otra parte se evitó el uso del tipo puntero, con el fin de que las rutinas puedan ser invocadas desde programas escritos en otros lenguajes que no dispongan de este tipo de datos.

Fisicamente una celda esta constituída por un bloque de locaciones contiguas de almacenamiento, contenidas en un pool. La dirección de una celda es la posicion de su primera locación dentro del pool.



El tamaño de cada locación es fijo, pero el número de locaciones de cada celda es variable. En nuestra implementación cada campo de la celda ocupa una locación.

Las celdas son creados dinamicamente desde un espacio de almacenamiento libre, a medida que son requeridos. El pool está constituído tanto por las locaciones libres como por las que están asociadas a celdas.



El hecho de que en un mismo espacio residan celdas y locaciones libres y que las locaciones no sean de un tipo de dato fijo, impide que el pool pueda ser accedido secuencialmente.

Cuando se elimina una celda sus locaciones son devueltas al espacio disponible.

- El espacio de almacenamiento de información - pool - tiene que ser capaz de almacenar valores de cualquiera de los dominios inherentes al problema, además de punteros a celdas y a descriptores.

Como en Pascal una variable solo puede ser capaz de almacenar valores de un único tipo, se convino en que la unidad de almacenamiento sea un registro variante con un único campo que puede ser de cualquiera de los dominios indicados en el párrafo anterior.

El pool esta implementado entonces por medio de un arreglo de registros variantes sin etiqueta, consistente cada uno de ellos de un único campo. La variación entre un registro y otro no es entonces el número de campos, sino el tipo del campo.

El control del tipo del contenido de la celda se realiza por programa, es decir, el registro no contiene un campo etiqueta. De hecho cada celda tiene asociado un descriptor que determina univocamente el tipo de cada campo de la celda.

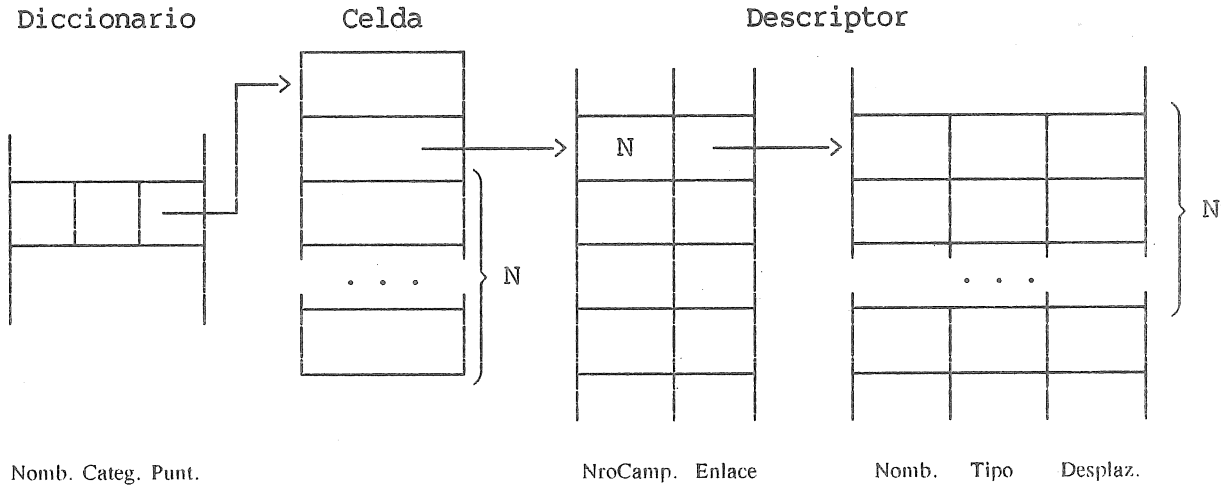
El pool contiene parte de la información estructural. Esto es, las celdas contienen un puntero a un descriptor y al menos un puntero a otra celda - la próximo de la lista.

- El diccionario, es una tabla con organización hash interno con resolución lineal de colisiones, constituída por ternas cuyas componentes contienen un nombre, una dirección y una indicación de si se trata de un nombre de descriptor o de celda.

La implementación de la tabla se realiza a través de un arreglo de registros, cada uno de los cuales consta de tres campos capaces de contener la terna que identifica a una entrada.

- Los descriptores estan distribuídos en dos tablas. Esto se debe a que cada descriptor contiene un número que indica la cantidad de campos que contiene la celda y una terna nombre-tipo-posición por cada campo de la celda.

Cada una de las tablas esta implementada a traves de un arreglo de registros. Una de las tablas contiene registros de dos campos, el primero indica la cantidad de campos de la celda, por ejemplo n, mientras que el segundo es un puntero al primero de los n registros de la segunda tabla, que permiten describir a los n campos de la celda. La segunda tabla contiene registros con un campo para cada componente de la terna nombre-tipo-posición que describe a un campo. La descripción de los n campos de una celda reside en n registros contiguos en la tabla.



- En nuestra implementación todo el área de almacenamiento es estático. La cabeza de la estructura contiene entonces los siguientes campos:

a) Total de Espacio Utilizado en el Pool

b) Total de Espacio Libre en el Pool

c) Puntero a la Primera Locación Libre en el Pool

d) Puntero a la Ultima Locación Libre en el Pool

e) Puntero a la Primera Locación Libre en la Tabla de Descriptores de celdas

f) Puntero a la Primera Locación Libre en la Tabla de Descriptores de Campos

g) Punteros a cada una de las listas de celdas.

2.0 Aplicaciones de la Estructura Plex

Una de las limitaciones de los simuladores de procesos actuales, es el hecho de que los datos que modelan un proceso, residen en estructuras de datos definidas a partir de constructores fijos, por ejemplo arreglos dimensionados. Esto provoca cierta inflexibilidad, ya que requiere que el diseñador del sistema conozca toda la información que será manipulada durante la simulación. Cualquier modificación del proceso provoca alteraciones en el diseño o al menos en las estructuras de datos [Ev 77].

La estructura de datos Plex surge como alternativa al uso de estructuras fijas.

La estructura de datos Plex provee entonces una forma elegante de representar los datos que describen un simulador de "flow-sheet".

El "flow-sheet" es representado por la estructura Plex en sus aspectos espaciales y temporales - topología y variaciones de su contenido en el proceso que se modela -, incrementándose su flexibilidad debido a que es posible modificar la topología sin trabajo extra. El concepto de estructura Plex ha sido incorporado en paquetes de simulación como el ASPEN, ampliamente usados en la industria petroquímica.

Los conceptos que importan a su definición la hacen sumamente útil y versátil como para extender el rango de su uso a otro tipo de simulaciones en donde estén involucradas transacciones de algún tipo entre objetos. Ejemplo clásicos pueden ser los circuitos eléctricos, las redes digitales y redes de computadoras.

La ventaja fundamental de esta estructura respecto a cualquier estructura fija es su flexibilidad, ya que es posible alterar la configuración de los datos sin alterar la estructura abstracta de los mismos. Además el espacio inutilizado disminuye, al tiempo que desaparecen las restricciones impuestas por una implementación utilizando constructores de datos fijos.

La implementación de Plex como tipo de dato abstracto y el encapsulamiento de la información que de ella se deriva, hacen que solamente pueda ser accedida por el usuario a traves de rutinas "ad-hoc" y permita establecer cierto tipo de protección sobre los datos con verificación en los derechos de acceso. La organización de los datos es asi transparente para el programador de aplicación, incrementando la modularidad de sus programas.

La forma de representación de la información, asi como su recuperación puede derivar en una base de datos que tanto sirve para la simulación en si misma como para la documentación o fuente de consulta, con todas sus características de flexibilidad y modularidad.

Este tipo de estructura constituye una buena base para próximos proyectos sobre diseño asistido por computadora. Esta última sería la linea de futuros trabajos en donde la estructura Plex juega un rol importante en cuanto al soporte de todo el sistema pero no es el fin último.

Apenice I

RUTINAS PARA MANEJO DE PLEX

1. Inicializa (EspLibre, Nomb, Plex)

   Inicializa:

   a) La cabeza de la estructura, dandole un valor nulo a los punteros a listas, uno a los punteros a locaciones libres en las tablas e insertando el nombre de la estructura y su espacio requerido.

   b) La tabla de descriptores con valores definidos como nulos.

   c) El diccionario con valores definidos como nulos.

   d) El pool, enlazando cada locacion con la siguiente, excepto la última donde se inserta nulo.

2. RequerirEspacio (Nb, N, Error, Plex)

   Este procedimiento toma N locaciones contiguas del espacio disponible, devuelve en Nb un puntero a la primera de ellas y actualiza el puntero a la primera locación libre en la cabeza.

   Como no todo el espacio disponible es contiguo, el procedimiento recorre la lista de locaciones libres, buscando las primeras N contiguas.

3. DevolverEspacio (Nb, N, Plex)

   No en todas las aplicaciones tiene sentido la eliminación de celdas, pero si esta acción fuera requerida, las N locaciones que ocupaba a partir de Nb son devueltas al final de la lista de locaciones libres en el pool.

4. InsertarDiccionario (Nomb, Pn, CodEnt, Error, Plex)

Inserta en el diccionario una entrada con la terna dada por los tres primeros parámetros, excepto si ya existía una terna con el primer valor igual a Nomb, en cuyo caso devuelve un código de error en Error.

La elección de la locación en el diccionario se realiza aplicando un función cuyo dominio son nombres y el rango son las locaciones del diccionario, si ya estaba ocupada se busca linealmente una locación libre.

5. RecuperarEntradaDiccionario (Nomb, Punt, CodEnt, Plex)

Devuelve en Punt y CodEnt los valores de la entrada en el diccionario asociada al nombre Nomb.

6. EliminarEntradaDiccionario (Nomb, CodEnt, Error, Plex)

Elimina, si existe, la entrada en el diccionario del tipo de CodEnt con nombre Nomb, insertando en su lugar un código de borrado.

7. CrearDescriptor (Nombre, NroCamp, Pd, Error, Plex)

Crea un descriptor llamado Nombre con NroCamp campos, devuelve en Pd un puntero al descriptor creado.

Actualiza los campos de la cabeza.

8. DefinirCampoDescriptor (Nomb, TipoCamp, K, Pd, Plex)

Dado el descriptor apuntado por Pd, define su K-esimo campo llamándolo Nomb y con tipo TipoCamp

9. CrearBead (Nb, NombDesc, NombBead, Error, Plex)

Recupera, si existe, el descriptor deseado a partir de la búsqueda de su nombre en el diccionario.

Reserva el espacio necesario en el pool, de acuerdo a lo indicado por el descriptor.

Inicializa el primer y segundo campo de la celda, con el puntero a la próxima celda y al descriptor, respectivamente.

Inserta el nombre de la celda en el diccionario.

10. RecuperarPunteroBead (NombBead, Nb, Error, Plex)

Dado un nombre, recupera si existe, el puntero a la celda asociada a ese nombre. Esta rutina no se ocupa de la categoria de la celda.

11. RecuperarPunteroDesc (NombDesc, Nd, Error, Plex)

    Devuelve en Nd en puntero al descriptor cuyo nombre es NombDesc, a partir de una búsqueda en el diccionario.

Notas :

1. El paquete se completa con rutinas que sirven como vinculo entre el programador y la estructura Plex, que dependen de la aplicación y que son las que en definitiva van a ser invocadas por los programas desarrollados por el usuario. Por ejemplo los programas de aplicacion no invocaran a CrearBead sino que dispondrá de una rutina por cada categoria, que ademas de crear la celda, la insertará en la lista adecuada.

2. El parámetro Plex incluye la totalidad de la estructura, esto es, cabeza, pool, diccionario y descriptores.

BIBLIOGRAFIA

[Ah 83] Data Structures and Algorithms
Aho A.V., Hopcroft J.E. and Ullman, J.D.
Addison-Wesley 1983

[Ho 76] Fundamentals of Data Structures using Pascal
Horowitz E. and Sahni S.
Computer Science Press 1976

[St 80] Data Structures Techniques
Standish T.A.
Addison-Wesley 1980

[Te 81] Data Structures using Pascal
Tenenbaum A., Augenstein M.
Prentice-Hall 1981

[Ev 77] System Structures for Process Simulation
Evans L.B., Joseph B., Seider W.D.
AIChe Journal (Vol.23,No.5,Septiembre 1977)