Estructuras enlazadas
Una aproximación independiente de la implementación

Perla Señas *Alejandro J. García*

Cátedra Estructura de Datos y Algoritmos
Departamento de Ciencias de la Computación
Instituto de Ciencias de la Computación
Universidad Nacional del Sur.
Av. Alem 1253 (8000) Bahía Blanca ARGENTINA
ccsenas@criba.edu.ar ccgarcia@criba.edu.ar

Palabras claves: estructuras enlazadas, estructuras dinámicas, lenguaje de diseño.

Resumen

El objetivo de este trabajo es presentar una propuesta para la enseñanza de las estructuras enlazadas en forma independiente de su implementación, evitando así generar confusión entre los diferentes conceptos asociados a este tema. La intención de la propuesta es separar claramente los conceptos de estructuras enlazadas y su necesidad como herramienta de representación de información, de las representaciones particulares con punteros o con cursores. Para ello se definirán primitivas en lenguaje de diseño para el manejo de estructuras enlazadas que permitirán abordar su estudio en una forma totalmente independiente de su implementación. Luego se presentarán los punteros y los cursores como una herramienta para la creación de estructuras enlazadas. Por último se definirá un ambiente de trabajo para los cursores, para que su manejo sea análogo al que se hace con el tipo de dato puntero.

Estructuras enlazadas
Una aproximación independiente de la implementación

1. Introducción

Todo curriculum de Ciencias de la Computación debe incluir un curso de Estructura de Datos y Algoritmos. Habitualmente la cantidad de temas incluidos en este curso supera los que pueden tratarse adecuadamente en el tiempo disponible. En la mayoría de los casos debe realizarse una selección de los temas con el riesgo de abandonar algunos importantes. Una alternativa interesante consiste en encontrar un enfoque que permita abordarlos en el menor tiempo posible, sin que esto implique una pérdida en la calidad del aprendizaje.

Los conceptos de estructuras enlazadas y estructuras dinámicas son temas centrales dentro de un curso de Estructura de Datos y Algoritmos. Lamentablemente muchas veces la presentación de las estructuras enlazadas se realiza a través de casos particulares (como listas, árboles, etc.) y utilizando una implementación particular como pueden ser los punteros. Esto en la mayoría de los casos tiende a generar confusión entre los conceptos: estructuras enlazadas, estructuras dinámicas y punteros. Por otro lado, la presentación de cursores como una forma de implementar enlaces, no suele hacerse en forma explícita, lo cual impide que se vea con claridad la diferencia entre las estructuras enlazadas como estructura de datos, y los cursores como una herramienta particular para implementarlas.

El objetivo de este trabajo es presentar una propuesta para la enseñanza de las estructuras enlazadas en forma independiente de su implementación, evitando así generar confusión entre los diferentes conceptos asociados a este tema. La intención de la propuesta es separar claramente los conceptos de estructuras enlazadas y su necesidad como herramienta de representación de información, de las representaciones particulares con punteros o con cursores.

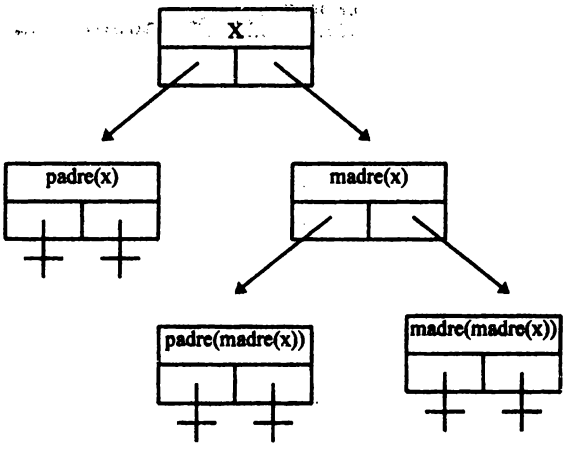
A fin de lograr los objetivos anteriores, se plantea la siguiente propuesta:

1. Presentar la necesidad de tener estructuras enlazadas a través de problemas donde resulta imperiosa su utilización. Por ejemplo la representación de datos definidos en forma recursiva, o como una estructura adecuada para obtener operaciones de búsqueda y/o actualización con menor tiempo de ejecución asociado.
2. Definir primitivas en lenguaje de diseño para el manejo de estructuras enlazadas, y de esta manera poder abordar su estudio en una forma totalmente independiente de su implementación. Así podrán construirse operaciones para los ejemplos antes mencionados, utilizando estas primitivas.
3. Presentar a los punteros simplemente como una herramienta para la implementación de estructuras enlazadas.
4. Presentar a los cursores como otra alternativa de implementación de estructuras enlazadas, y proveer un entorno que permita utilizarlos en forma análoga a como se utilizan los punteros.

2. Ejemplo para la plantear la necesidad de estructuras enlazadas.

Una forma de introducir las estructuras enlazadas es presentando problemas cuya resolución intuitiva cree la necesidad de estructuras de este tipo. De esta forma, el alumno verá como natural su incorporación como herramienta de representación de información.

Un ejemplo típico con las características anteriores, es la representación de la genealogía de una persona, la cual condensa la información sobre sus ancestros. La genealogía se define recursivamente mediante el nombre de la persona, la genealogía de la madre (si se conoce) y la genealogía del padre (si se conoce). Una estructura adecuada para representar este problema (para una persona *x*), puede representarse gráficamente como sigue, si se tiene en cuenta que no se conoce de antemano el nivel de anidamiento de la misma:



La definición de esta estructura es netamente recursiva y está dada por:

Genealogía
- i) **Registro con**
  - nombre de la persona : cadena de caracteres
  - padre: genealogía
  - madre: genealogía , si el nombre de la persona es conocido.
- ii) **nulo** , en caso contrario.

A continuación se describirán primitivas para representar y manejar esta estructura en lenguaje de diseño.

3. Primitivas de lenguaje de diseño para manejo de estructuras enlazadas.

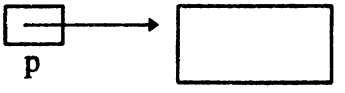
La utilización de un lenguaje de diseño de algoritmos en los cursos iniciales de programación es una alternativa interesante para poder introducir conceptos, sin necesidad de

trabajar con el nivel de detalle de un lenguaje de programación. Esta idea puede extenderse y ser empleada para la presentación de temas más complejos en cursos superiores.

Habitualmente las estructuras enlazadas se presentan como conglomerados de unidades básicas llamadas *celdas*, vinculadas entre sí a través de *enlaces* (direcciones de celdas). Las celdas pueden contener diferentes datos, y en particular algunos de los datos pueden ser enlaces a otras celdas. Esto es lo que permite la creación de estructuras enlazadas complejas.

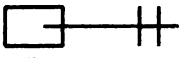
Se describirán a continuación los elementos necesarios para expresar en lenguaje de diseño, las primitivas para el manejo de estructuras enlazadas.

- El lenguaje de diseño tendrá dos tipos de objetos nuevos: los *enlaces* y las *celdas*. Si *p* es un enlace, entonces se notará con *celda(p)* a la celda apuntada por *p*. Gráficamente se tiene:



celda(p)

- Decir que *p* es un enlace a *celda(p)* es equivalente a decir que *p* tiene la dirección de *celda(p)*.
- Es necesario también introducir un valor de enlace nulo (que no apunta a ninguna celda) al cual se lo representará con la constante *nulo*. Si *p* = *nulo*, entonces *celda(p)* no existe. Gráficamente:



- En el caso que una variable de enlace *p* no tenga asociada ninguna celda, *p* tendrá el valor *nulo*.
- Para la generación dinámica de las celdas se introduce la operación *CrearCelda(p)* donde *p* es una variable de enlace. El efecto de ejecutar *CrearCelda(p)* es la creación de una nueva celda que se identifica con *celda(p)*. Además *p* toma como valor la dirección de *celda(p)*.
- Para que una celda que ha sido creada previamente deje de existir se introduce la operación *EliminarCelda(p)*, donde *p* es una variable de enlace. La ejecución de esta operación hace que *celda(p)* deje de existir, dejando a *p* con un valor *nulo*.

Siguiendo con el ejemplo anterior una genealogía se representará en este lenguaje de la siguiente forma:

```
Genealogia = enlace a Celda
Celda = Registro con
            persona: cadena de caracteres
            padre: enlace a Celda
            madre: enlace a Celda
```

Obsérvese que la constante de enlace *nulo* resuelve el caso trivial de la definición recursiva.

A continuación se presenta un algoritmo en lenguaje de diseño para la generación de la genealogía de una persona.

```
Algoritmo: Crear Genealogía
Dato de Entrada: Nombre : Cadena de caracteres
Dato de Salida: p : Genealogia
Comienzo
        CrearCelda(p)
        Celda(p).persona ← Nombre

        Si ConozcoMadre(Nombre,NombreMadre)
        entonces Celda(p).madre ← CrearGenealogia(NombreMadre)
        sino Celda(p).madre ← nulo

        Si ConozcoPadre(Nombre,NombrePadre)
        entonces Celda(p).padre ← CrearGenealogia(NombrePadre)
        Sino Celda(p).padre ← nulo

        DEVOLVER p

Fin DE Crear Genealogía
```

Donde ConozcoMadre(X,Y) devuelve verdadero si se conoce el nombre de la madre de X, el cuál es devuelto en Y. Lo mismo vale para ConozcoPadre(X,Y).

Es importante destacar que el espíritu de este trabajo es presentar una metodología para la enseñanza de estructuras enlazadas, a través de un lenguaje de diseño. No está dentro de los objetivos de este trabajo la imposición de una notación estándar para el lenguaje de diseño. Lo importante aquí es la utilización de un lenguaje de diseño, y no los detalles de su notación en sí. Cada profesor evaluará cuál es la notación más conveniente, dependiendo de factores tales como: notación utilizada en cursos anteriores, lenguaje de programación a utilizar posteriormente, etc..

4. Implementación de las primitivas.

Se describirá a continuación la implementación de las primitivas del lenguaje de diseño para el manejo de estructuras enlazadas.

4.1. Usando punteros.

En lenguajes de programación de alto nivel que disponen del tipo puntero, la correspondencia con el lenguaje de diseño propuesto es casi directa ya que habitualmente poseen procedimientos estándar para la creación y eliminación de variables dinámicas. El siguiente cuadro muestra la equivalencia entre el lenguaje de diseño propuesto, Pascal y Modula-2.

Lenguaje de diseño	Pascal	Modula-2
p:**enlace a** Celda	p:^TipoCelda[1]	p:POINTER TO TipoCelda
celda(p)	p^	p^
CrearCelda(p)	new(p)	Allocate(p,SIZE(p^))
EliminarCelda(p)	dispose(p)	Deallocate(p, SIZE(p^))
nulo	nil	NIL

4.2. Usando cursores.

Cuando la implementación de estructuras enlazadas se hace usando un lenguaje que no dispone de punteros, debe recurrirse (en forma explícita o no) al uso de cursores. Una forma ordenada y efectiva para trabajar con cursores consiste en la creación de un ambiente donde pueda desarrollarse lo presentado en el lenguaje de diseño anterior.

El ambiente consistirá de:

- un espacio de trabajo para alojar celdas.
- variables de tipo enlace que serán direcciones del espacio de trabajo.
- un método para mantener la información del lugar disponible en el espacio de trabajo.
- operaciones para la creación y eliminación de celdas, que en este caso corresponderán a solicitar o devolver celdas al espacio de trabajo.

Implementación del ambiente de trabajo.

- El espacio de trabajo será un arreglo unidimensional de celdas.
- Las celdas serán estructuras con campos de información donde al menos un campo será de tipo enlace. A dicho campo lo llamaremos genéricamente campo *siguiente*.
- La constante *nulo* tendrá el valor 0.
- El valor de un enlace será un número entre 0 y el índice mayor del espacio de trabajo.
- Para mantener la información de las celdas que están disponibles, se mantendrán todas las celdas libres enlazadas por uno de sus campos de enlace, en una estructura lineal. Se tendrá una variable de tipo enlace, que llamaremos *CeldasLibres*, la cual contendrá la dirección de la primer celda disponible de dicha estructura. Será necesario un procedimiento inicializar que se encargará de enlazar todas las celdas disponibles y de dar el valor inicial a la variable *CeldasLibres*.

De esta manera, se definen las siguientes estructuras y operaciones:

Constantes

```
nulo = 0
FinEspacio = (un valor adecuado)
```

[1] TipoCelda es un identificador de tipo declarado previamente.

- Al implementarse una estructura enlazada con punteros se deberá atender aquellas situaciones donde el lenguaje de diseño asigna valor nulo a los enlaces. Puede ocurrir que en la definición del lenguaje de programación en las situaciones correspondientes, el valor asociado al puntero sea indeterminado. De ser así, es recomendable que para tales situaciones se considere explícitamente la asignación de la constante nula a la variable puntero.

5. Conclusiones

La presentación de estructuras enlazadas propuesta y la creación de un ambiente de trabajo para su implementación con cursores, permite estudiar las distintas estructuras de datos en un nivel de abstracción más alto, sin tener que preocuparse por detalles de implementación. Dichos detalles pueden resolverse luego en el desarrollo de los ejercicios prácticos. De esta manera es posible desarrollar los temas en menor tiempo, logrando además una mayor claridad en el aprendizaje de los conceptos involucrados.

6. Referencias

ACM/IEEE-CS Joint Curriculum Task Force. Computing Curricula 1991.New York, ACM Press,1991

Aho,Alfred; Hopcroft, Jhon an Ullman, Jeffrey. Estructuras de datos y algoritmos. Adisson-Wesley. 1988

Collins, William J. Data Structures. Adisson-Wesley. 1992

Horowitz, Ellis and Sartaj, Sahni. Data Structures in Pascal. New York. W. H. Freeman and Company. 1990

Jarc, Duane. Data Structures: A unified view. ACM SIGCSE VOL.26 . June 1994.

Muller, Hausi A. Diferences between Modula-2 and Pascal. ACM SIGPLAN. V19. Oct. 1984.

Wirth, Niklaus. Programming in Modula-2. Springer-Verlag. 1983.