Escuela: EPET N° 1 de Caucete.

Docente: Luis Sesé.

Grado: 5to año, ciclo orientado.

Turno: Mañana.

Área Curricular: Conversión y Reparación de Datos.

Título de la propuesta: Estructura de la información.

En su aspecto más general, el arte de la programación de ordenadores tiene mucho que ver con eso que llamamos la "inteligencia" humana; los procesos de abstracción; la capacidad de imaginar un problema y sus posibles soluciones (hemos dicho que programar es formular la solución de un problema). El proceso suele ir de lo general a lo particular y viceversa (de las ideas abstractas a los detalles concretos), en un proceso que se realimenta constantemente.

En una primera fase, el programador concibe una imagen del problema (el primer paso es "comprender el problema"), sin embargo, para formular la solución, se necesita un vehículo o soporte sobre el que construirla. En nuestro caso conocer un lenguaje de programación, lo que en el fondo tiene un doble sentido: el primero y más importante, es conocer que herramientas ofrece. Estas herramientas funcionan como ideas-soporte; como ladrillos con los que construir la solución como si fuese una construcción material. Esto significa, sobre todo, que el programador piensa en función de las herramientas disponibles. Sin ellas es incapaz de imaginarse ninguna solución concreta. El segundo significado es mucho menos importante (conceptualmente hablando); supone el conocimiento de una serie de reglas formales de utilización de los elementos del lenguaje. Sería la fase de mera codificación, tarea para la que existen herramientas cada vez más eficaces. Por ejemplo, la amplia colección de herramientas RAD que componen las modernas "suites" de desarrollo.

La situación es comparable a la de un hipotético Robinsón que en una isla desierta tuviese que resolver el problema de su alojamiento. Tiene conciencia del problema y de su solución en abstracto: "un refugio". Sin embargo, para pensar algo concreto necesita antes explorar la isla, conocer que posibilidades ofrece. ¿Existen cuevas naturales?, ¿Madera?, ¿Restos de naufragios? ¿Algo que pueda adaptarse? etc. Evidentemente la forma mental de su

refugio dependerá de los medios disponibles; posteriormente, la construcción en sí, será cuestión de detalle y de su propia habilidad manual.

Datos y algoritmos

"Muéstreme su código y esconda sus estructuras de datos, y continuaré intrigado. Muéstreme sus estructuras de datos y generalmente no necesitaré ver su código; resultará evidente". Eric S. Raymond. “La Catedral y el Bazar".

En informática, las ideas-soporte a que aludíamos son principalmente de dos tipos: Relativas a la información y a su manipulación (los datos y los algoritmos respectivamente).

En lo que concierne a los primeros (los datos), existen multitud de formas de organizarlos; a estas formas las denominamos estructuras de datos. En cuanto a los segundos, existen un número casi infinito de algoritmos, pero en lo que aquí nos ocupa, tienen especial importancia los relacionados con el acceso a la información, es decir, los relacionados con su almacenamiento y recuperación. Algoritmos que Shildt denomina mecanismos de datos. Generalmente se considera que estos mecanismos realizan tres tipos de operaciones: Inserción, borrado y búsqueda de la información en la estructura correspondiente.

Nota: La denominada STL ("Standar Template Library"), que es parte esencial de la Librería C++ Estándar es en realidad un conjunto fascinante de estructuras de datos (aquí llamados contenedores) y de algoritmos para su manejo. Con su ayuda se han construido las aplicaciones más grandes y exigentes que se hayan utilizado nunca en informática.

Las estructuras y mecanismos de datos han sido muy estudiados; constituyen en sí mismos dos mundos dentro de la informática, y dado que son las herramientas para construir un programa, además de la experiencia, es conveniente disponer del conjunto más completo posible de ellas (en este aspecto la programación se parece al bricolaje). Por ejemplo, si nos referimos a los algoritmos, los métodos de ordenación ("Sort") o de construcción de índices (que viene a ser equivalente), han sido muy estudiados, de forma que se conoce cuáles son los más eficientes para cada caso (algunos incluso patentados). Si el lenguaje no los tiene pre construidos, tendremos que fabricarlos manualmente con los medios disponibles. En este aspecto, un lenguaje es tanto mejor cuanto más abstractas sean las herramientas que nos ofrezca (estructuras y mecanismos de datos). Lo que significa que

estarán más cercanas a la idea que tiene en mente el programador-humano y más alejadas de la forma concreta que adoptarán en la máquina. Este es precisamente el significado de la expresión "Lenguaje de alto o bajo nivel", el mayor o menor grado de proximidad que ofrezcan sus herramientas con las ideas (abstracciones) en la mente humana.

Al abordar un problema y decidirse por una estructura o mecanismo de datos concreto, hay un aspecto importante que debe ser conocido y tenido en cuenta por el programador. Es el hecho de que unas estructuras son más adecuadas para determinados mecanismos que otras. Dicho en otras palabras: las relaciones entre las estructuras de datos y los algoritmos que las manipulan son muy significativas. Por ejemplo: determinadas estructuras son muy adecuadas para operaciones de inserción y recuperación, de forma que los algoritmos que realizan estas operaciones son muy simples y rápidos. En cambio, estas mismas estructuras pueden ser inadecuadas para buscar información en su interior. Resulta, por tanto, que la relación de cada tipo de estructura de datos con los mecanismos básicos, inserción, borrado y búsqueda, es muy diferente, y debemos pensar detenidamente qué relación tiene nuestra estructura con los mecanismos básicos (como la usaremos).

Tanto las estructuras como los mecanismos de datos son importantes. De hecho, en el diseño de programas caben dos enfoques, según las consideraciones iniciales consideren prioritarias a unas u otros. En algunos casos la elección de la estructura de datos es el primer considerando del diseño, ya que la experiencia ha indicado que el trabajo total de desarrollo y la calidad del resultado dependen grandemente de la idoneidad de las estructuras utilizadas. Una vez elegidas, la elección del mecanismo es una consecuencia directa de la decisión anterior. En otros casos la línea de razonamiento sigue el camino inverso; se deciden las estructuras porque ciertas operaciones críticas utilizan algoritmos que funcionan mejor en esas estructuras.

La preponderancia de uno u otro criterio se deja sentir incluso en el diseño de los lenguajes de programación e incluso de los sistemas operativos. Los lenguajes orientados a objetos (como C++) están diseñados alrededor del concepto estructura de los datos, mientras en otros, por ejemplo, Lisp, el diseño se centra en el aspecto algorítmico. Al tratar de las plantillas ("Templates"), se comenta más ampliamente la relación entre datos y algoritmos.

Nota: Tanto en la programación procedural (tradicional) como en la POO, diseñar un programa supone construir un modelo. Este modelo goza de la propiedad de ser

"ejecutable", mostrando un comportamiento distinto en cada ocasión según las condiciones de partida o las suministradas durante la ejecución. El proceso de diseño comienza identificando ciertos elementos que son considerados piezas elementales del modelo, a los que denominamos "Datos". A estos datos se les añade cierta funcionalidad que les es propia. En la POO, este proceso conduce a la definición de las clases que compondrán el modelo. El resto del diseño consiste en establecer las relaciones de estos elementos individuales entre sí, y con respecto a las condiciones de entrada.

Estructura Física y estructura Lógica

Al tratar de las estructuras y contenedores de datos, es conveniente tener algunas ideas básicas claras y precisas sobre los mecanismos involucrados.

La estructura lógica se corresponde con la idea que en principio tiene el programador sobre cómo están organizados los datos, y coincide aproximadamente con la forma en que son manipulados los datos por el programa de alto nivel.

En la concepción de la estructura lógica, el programador puede razonar más o menos en los siguientes términos: "Voy a crear un fichero de clientes donde los datos de cada cliente estarán agrupados en un registro. Posteriormente accederé los registros por número de cliente (que será único) o por nombre, para lo que estarán ordenados alfabéticamente (construiré un índice con el código de cliente y otro de nombres) ...". Si está habituado a la programación de bases de datos con herramientas de alto nivel, quizás su razonamiento sea el siguiente: "Voy a crear una tabla de clientes donde incluiré los datos de cada cliente, comenzando por una columna para el código que será el índice principal (será un INT UNSIGNED). También crearé un campo "nombre" que será un VCHAR NOT NULL UNIQUE...".

En uno u otro caso, la estructura (se llame "fichero" o "tabla") es una unidad lógica que se compone una multitud de elementos individuales (se llamen "registros" o "filas" -según la cultura del programador-). La estructura así concebida tiene un orden, ya que sus elementos estarán conceptualmente uno detrás de otro. Este orden será numérico, si el acceso se realiza por código de cliente, o alfabético de nombres si el acceso se realiza por nombre. A su vez, esta estructura lógica se divide aún más finamente: cada elemento se puede considerar dividido en multitud de campos. Aparte de los ya mencionados para código de cliente y nombre, pueden existir muchos más: dirección, teléfono, saldo, clasificación financiera, fecha última compra, vendedor asignado, etc. etc.

Por su parte, la estructura física corresponde a la forma en que están contenidos los datos en la máquina, de la que existen dos versiones: una corresponde a la que adoptan los datos en memoria; la otra a su almacenamiento externo (disco). Ambos esquemas son distintos.

Resulta evidente que la estructura física de datos en los almacenamientos externos no se corresponde exactamente con estructura lógica. En principio, el fichero o tabla de clientes antes mencionado, puede estar representado físicamente por varios ficheros que pueden ser multi-volumen. Es decir: ocupar más de un volumen lógico en la máquina que los alberga. Si son aplicaciones de red, pueden estar incluso en máquinas remotas, distintas de la que ejecuta la aplicación. Además, aunque nos figuramos la estructura lógica es un todo continuo (suponemos que después de un cliente sigue otro), sabemos que la estructura física correspondiente, incluso si se trata solo de un fichero, está compuesta por trozos "clusters" que pueden estar dispersos en el disco.

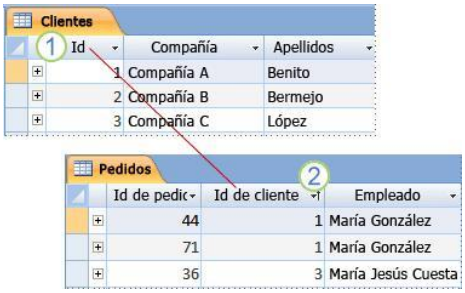
La estructura lógica está ordenada (por números o por nombres en nuestro ejemplo). En cambio, la estructura física puede estar construida simplemente por el orden "natural" es decir, de creación de los propios registros. Generalmente, la "apariencia" de ordenación es el resultado de un proceso complejo que utiliza índices, tablas y punteros, para proporcionarnos un acceso ordenado a una estructura mucho más caótica.

Como queda dicho, los datos son manejados por el programador y el programa (que es la expresión concreta de las ideas de aquel) en términos de esta estructura lógica. En lo tocante a este aspecto, las herramientas que ofrezca el lenguaje o entorno de programación, serán de mayor nivel cuanto mayor sea la distancia con que pueda ser manejada la estructura lógica de datos respecto de su verdadera estructura física. Precisamente el manejo de tales estructuras ("Databases"), ha originado toda una rama de la industria del software que ha alcanzado un alto nivel de sofisticación y especialización. En proyectos grandes es usual manejar los datos a través de estas herramientas. En estos casos el lenguaje C++ las utilizaría a través de interfaces. Por ejemplo, interfaces con motores relacionales (SQL). Sin embargo, es rara la aplicación en que el programador no deba manejar uno o varios ficheritos (por ejemplo, con parámetros de configuración), a un nivel "relativamente" bajo, haciendo uso de las herramientas (relativamente simples) que ofrece la Librería Estándar. También es usual que deba manejar, también a bajo nivel, alguna estructura de datos en memoria, por ejemplo, una

matriz, mediante técnicas totalmente distintas de las que se utilizarían con un fichero de disco.

Como resumen, podemos afirmar que el programador, al menos el programador C++, puede concentrarse en la estructura lógica, pero sin olvidar de vez en cuando mirar la estructura física con el rabillo del ojo. Como hemos señalado antes, existen distintos tipos de estructuras de datos (lógicas y físicas) que se diferencian grandemente en su grado de adecuación a diversas formas de almacenamiento y recuperación de la información, por lo que es conveniente que el programador tenga ciertas nociones al respecto.

Primera actividad: Desarrolle un documento de texto donde indique el orden de las 3 imágenes, debatiendo el criterio de lo que considera desde una estructura de datos físicos hasta una estructura de datos lógicos. Justifique su respuesta.



Segunda actividad: Genere una lista en papel o computadora de los objetos que tiene contacto con sus manos desde que se levanta hasta que comienza a ingerir el desayuno.

Tercera actividad: Clasifique los elementos en una tabla con índice de sector ej:

Índice	Sector	Elementos
1	Pieza	Despertador, celular, ropa
2	Baño	Surtidor, cepillo, dentífrico

De esta manera tendremos la información organizada de forma lógica de los objetos que tenemos contactos antes de ingerir el desayuno diario.

Envié documento en Word con las 3 actividades a la casilla de correo de lasese@sanjuan.edu.ar – Cualquier consulta diríjase a la misma dirección.

Director: Mario Gomez.