Capítulo 12

Consejos y Reflexiones sobre Programación

En los siguientes párrafos pretendo condensar en una serie de consejos y reflexiones toda mi experiencia como programador y como profesor de Lógica de Programación. Algunos de estos comentarios obedecen a criterios puramente técnicos y otros sencillamente son recomendaciones que me han hecho mucho mas fácil el trabajo de la programación y su correspondiente convivencia con diferentes lenguajes de computador. Por esta razón espero que este capítulo sea para usted un aporte significativo en su tarea de formación como Programador.

Acerca de la Lógica

Siempre que usted vaya a resolver un problema sea muy lógico, esto quiere decir que sencillamente guíese por sus mínimos razonamientos y busque siempre el camino mas obvio y sencillo. No existe un problema que se resuelva con lógica cuya solución no sea sencilla. Antes de comenzar a pensar en la lógica de programación piense en su propia lógica. Diseñe las soluciones pensando en sus propias reglas y luego si ajústese a las reglas que la lógica de programación impone para facilitar la posterior codificación.

Es muy importante que usted poco a poco destine parte de su tiempo a resolver problemas, así no sean estrictamente de programación, dado que en esos momentos en donde usted se sienta a pensar detenidamente en la búsqueda de solución de un determinado problema, en esos momentos es donde usted realmente está utilizando su cerebro. Normalmente nuestro cerebro se va acostumbrando, y así es como lo orienta la educación formal, a buscar entre sus conocimientos las soluciones que se estén necesitando pero muchas veces tenemos que crear soluciones y es allí en donde nos encontramos que son muy pocas las veces en las que ponemos a funcionar nuestro cerebro.

Yo espero que usted, amigo lector, no se me vaya a ofender ya que no es el propósito de este párrafo incomodarlo lo que si quiero es que usted piense cuántas veces realmente se ha sentado a crear una solución de un problema y encontrará que son muy pocas las veces ya que en aquellas oportunidades en donde ha tratado de hacerlo y, por ventura, ha encontrado la solución es porque su cerebro ha buscado en su "biblioteca de conocimientos" alguna solución análoga a problemas parecidos y la ha "ajustado" al problema en mención.

Por tal motivo es muy importante que de vez en cuando resuelva acertijos matemáticos, problemas con palitos, dados, cartas e incluso hasta resolver adivinanzas. Este tipo de problemas le van permitiendo a usted buscar soluciones espontáneas, originales, creadas por usted mismo y que además son solución a un determinado problema planteado. Todos esos juegos de lógica que mas de una noche nos han puesto a pensar son los que van haciendo que el cerebro cree soluciones y no las busque en las que ya conoce. Es muy importante que tenga en cuenta todo esto dado que en programación usted va a necesitar crear soluciones a problemas determinados basadas en sus conceptos y en el conocimiento que tenga de las herramientas y de los conceptos aquí planteados.

Siempre que usted se enfrente a un problema, no lo olvide, busque el camino mas lógico para resolverlo. Cómo saber cuál es el camino mas lógico..? Pues sencillamente la solución mas obvia es la que demarca cuál es el camino mas lógico. Siempre busque la solución mas obvia antes de comenzar a aplicar teorías y conceptos como los planteados en este libro. La aplicación de dichas teorías y conceptos debe ser un paso posterior. Inicialmente lo que usted debe tener aproximadamente claro es un camino de solución y si se detiene a pensar en el problema no será raro que la mayoría de las veces tenga el camino mas obvio de solución a dicho problema.

La lógica es ese conjunto de razonamientos que nos permiten solucionar fácilmente determinados problemas o lograr fácilmente determinados objetivos. Cada persona puede tener un enfoque diferente en cuanto a dicha solución y es muy importante que, cuando se trabaja en equipo, escuchar cuál es la solución de los otros. Indiscutiblemente que para cada problema ha de existir una solución óptima, obvia y además muy sencilla. Porqué razón cada persona puede llegar a encontrar una solución diferente a un determinado problema..? Son múltiples las explicaciones pero se debe destacar dentro de ellas el entorno social, la preparación, el conocimiento, la convivencia y la utilización de conceptos nuevos acerca de la misma lógica, su mismo entorno personal y muchas mas razones que pueden hacer que una persona vea la solución de un problema con una óptica diferente a como la podemos ver nosotros.

Lo que para una persona es absolutamente ilógico para otra es completamente lógico y es posible que ambas tengan la razón (o al menos crean tenerla) dadas sus propias condiciones personales e intelectuales. Sin embargo podemos decir que si se mira un problema con una óptica aproximadamente neutral nos podremos aproximar a la solución mas sencilla y obvia. Cómo poder llegar a obtener una óptica aproximadamente neutral..? Considero que solo hay una forma de acercarse a este concepto y es estudiando conceptos que nos permitan lograr este objetivo. No es fácil determinar en qué momento hemos alcanzado una lógica aproximadamente normal pero

cuando el problema a solucionar puede ser resuelto con la utilización de la tecnología entonces ésta se convierte en el catalizador y en la regla de medida para saber hasta donde nuestra solución es realmente óptima o no.

No olvide que la lógica computacional le permitirá buscar soluciones que puedan ser implementables con tecnología. Por esta razón es que la buena utilización de la misma nos va a permitir saber hasta donde nos hemos acercado a la solución óptima. Me arriesgaría a decir que un problema que se solucione con lógica computacional solo tendrá una y solo una solución óptima. Es decir la solución mas sencilla de implementar, la mas obvia y la mas entendible, a la luz de los conceptos de la misma lógica computacional.

Acerca de la Metodología para solucionar un Problema

Es muy posible que a esta altura usted se haya dado cuenta de algo en cuanto a la metodología para solucionar un problema y es que esta Metodología no solo se refiere a problemas de la lógica computacional sino que involucra cualquier problema. Note usted que los tres pasos explicados en este libro son ni mas ni menos el secreto para que cualquier problema, sin importar de qué orden sea, tenga, mínimamente, una buena solución.

El objetivo, lo principal en todo lo que nos proponemos hacer en nuestras vidas, muchas empresas han fracasado solamente porque no tenían el objetivo suficientemente claro, muchos propósitos se han ido al piso solo porque el objetivo no era tan diáfano como se pensaba. En cuestión de lógica de programación el objetivo es lo principal. Recuerde “No de ningún paso hacia delante si no tiene exageradamente claro el objetivo”. Cuando usted tiene claro el objetivo entonces obtiene dos ganancias: Sabe claramente qué es lo que quiere hacer y sabe claramente hasta donde debe llegar o sea en donde debe parar. Pareciera que estas dos ganancias fueran lo mismo pero en realidad no lo son aunque no puedo negar que si están fuertemente conexas.

El objetivo marca el faro hacia el cual debemos enrutar todos nuestros esfuerzos y la clave para lograrlo es, precisamente, no perderlo de vista. Sin un objetivo suficientemente claro no podemos empezar a desarrollar nada pues es posible que estemos tomando un camino equivocado y no nos estemos dando cuenta. Cómo saber que el objetivo está realmente claro? Pues con el solo hecho de que usted ha conceptualizado perfectamente lo que quiere lograr, el hecho de que usted pueda explicar con absoluta seguridad qué es lo que pretende obtener, ya con eso usted podrá sentir que el objetivo está completamente claro.

No dé ni un solo paso hasta tanto no tenga una absoluta certeza de lo que usted quiere lograr. No avance porque es posible que llegue a avanzar por el camino equivocado y termine logrando algo muy distinto a lo que había sido su objetivo original. Primero que nada un objetivo claro y ahí si podrá usted pensar en todos los demás elementos que conforman esta metodología.

El objetivo es como ese farol que ilumina una calle, inicialmente, oscura. Cuando usted tiene bien claro el objetivo a lograr inmediatamente el camino para lograrlo automáticamente se aclara. Cuando a usted le dicen que debe ir hasta ese edificio azul que se ve al fondo inmediatamente

comienza a buscar todos los caminos para llegar a él y verá mas de uno que le permita lograr llegar hasta el edificio en mención. Eso es lo mismo que pasa cuando usted tiene un objetivo claro. Ese camino para lograr un objetivo es lo que se llama Algoritmo. El algoritmo es el conjunto de acciones (en el caso de un seudocódigo es el conjunto de órdenes) que nos permiten lograr un objetivo. Cuando hemos depurado nuestra lógica frente a la solución de varios problemas podemos decir que dicho Algoritmo se aproxima altamente a la solución óptima.

Cuando se trate de algoritmos computacionales deberán ir orientados a facilitar la codificación posterior precisamente para tener toda la certeza de que el computador va a ejecutar lo que nosotros hemos concebido como solución. Trate de utilizar en sus algoritmos, órdenes que sean de fácil conversión a cualquier lenguaje de programación convencional. No olvide que lo mas importante en el desarrollo de soluciones computacionales es la lógica que se haya utilizado para solucionar un determinado problema. Siempre tenga en cuenta que de una buena lógica utilizada en la construcción de los algoritmos dependen los buenos programas, es decir, las buenas aplicaciones.

Igualmente de nada sirve la presunción de que nuestros algoritmos están bien o sea de que cumplen los objetivos planteados si no nos lo demostramos a nosotros mismos haciendo unas buenas “pruebas de escritorio”. Siempre debe tener en cuenta que un algoritmo al que no se le ha hecho una rigurosa prueba de escritorio no es mas que un conjunto de órdenes posiblemente incoherente. Lo único que le da verdadero respaldo a un algoritmo es la prueba de escritorio que es la que nos permite saber si realmente el algoritmo está cumpliendo el objetivo o sino.

La prueba de escritorio nos brinda dos elementos que a mi juicio son lo mas importante en la construcción de los algoritmos: en primera instancia la prueba de escritorio nos va a permitir saber si realmente el algoritmo cumple el objetivo o no o sea, en términos coloquiales, la prueba de escritorio nos permite saber si el algoritmo está bien. En segundo lugar la prueba de escritorio nos va a permitir saber en dónde está el error o los errores de un determinado algoritmo para ser corregido, porque no solo se necesita saber si cumple o no con el objetivo un determinado algoritmo sino además saber en donde están las fallas y las razones por las cuales no está cumpliendo el objetivo.

Solo hasta cuando usted ha realizado una prueba de escritorio rigurosa y ha confrontado los resultados finales con el objetivo inicial solo hasta ese momento podrá realmente saber si su algoritmo estaba bien concebido o no. Cómo hacemos entonces para realizar una prueba de escritorio rigurosa..? Pues muy sencillo, escriba su algoritmo en formato de seudocódigo y coloque en el papel un rectángulo para lo que va a pasar en la pantalla y otro rectángulo para lo que va a pasar en la memoria. En el rectángulo de la memoria coloque las variables que va a utilizar su algoritmo y comience a ejecutar línea a línea cada orden escrita. No suponga nada. No se salte ninguna orden. No presuma ningún resultado. Usted tendrá que ejecutar el algoritmo tal y como el computador lo va a realizar. El no supone nada, no se salta ninguna línea ni presume ningún resultado.

Solo de esta forma podemos llegar a los mismos resultados que llegaría el computador. Esto quiere decir que si al realizar una prueba de escritorio vemos que nuestro algoritmo arroja unos resultados que no coinciden con el objetivo entonces si el computador ejecutara dicho algoritmo (convertido en programa por supuesto) también arrojaría resultados errados. Vaya utilizando las variables de memoria en la medida en que las va utilizando el algoritmo y cada vez que le coloque un nuevo valor a una variable tache el valor que tenía almacenado antes dicha variable. Sin suponer nada ejecuta una a una cada instrucción del algoritmo y cuando haya terminado mire lo que queda en la pantalla o mejor dicho en el rectángulo de la pantalla.

Si lo que usted obtiene allí coincide con el objetivo inicial que buscaba lograr entonces su algoritmo estará bien y podrá comenzar a codificarlo. Si usted nota que los resultados presentados en su "pantalla" no coinciden con los que inicialmente había planteado como objetivo entonces esto querrá decir que su algoritmo está mal y que por lo tanto usted deberá corregir los errores que la misma prueba de escritorio le indique en donde están. No olvide que en el instante de desarrollar una prueba de escritorio no se presume nada, no se supone nada ni se salta ninguna instrucción.

Podríamos concluir en este conjunto de reflexiones acerca de la metodología para solucionar un problema que de la realización eficiente de las pruebas de escritorio depende el tiempo que usted gane o pierda, de la buena construcción de algoritmos dependen las buenas aplicaciones de programación y de la claridad de los objetivos depende su prestigio como programador.

Acerca de las Variables y los Operadores

Cuando usted vaya a desarrollar un algoritmo no pierda mucho tiempo pensando en las variables que va a necesitar. Comience por declarar las variables que saltan a la vista. Por ejemplo si su algoritmo comienza diciendo *Leer un número entero y....* ya con eso es suficiente para que usted sepa que va a necesitar mínimamente una variable de tipo entero dado que al leer un número este debe quedar almacenado en algún lugar de memoria y ese lugar indiscutiblemente tiene que se una variable. Igualmente del mismo enunciado se puede concluir el tipo de dato que necesitamos.

Cuando vaya a iniciar un algoritmo declare las variables que saltan a la vista como cuando usted va a preparar un arroz con pollo salta a la vista que va a necesitar arroz y pollo pero también sabemos que no solo esos dos elementos forman parte de un arroz con pollo, también necesitaremos sal, condimentos y otros productos que no saltan a la vista de manera tan obvia como el arroz y el pollo. Entonces inicialmente declare las variables que saltan a la vista en el mismo enunciado, no se detenga a pensar en las demás variables que va a necesitar pues éstas van surgiendo en la medida que las va necesitando en el desarrollo del mismo algoritmo.

En este libro se han explicado y se han utilizado tres tipos de datos: los datos de tipo entero, los datos de tipo carácter y los datos de tipo real. Cada uno tiene unas características técnicas que permiten manipular y manejar variables con ese tipo de datos. Sin embargo tenga en cuenta que muchos lenguajes de programación cuentan con tipos de datos diferentes adicionales a los aquí explicados y en mas de una oportunidad esos tipos de datos nos pueden facilitar de una manera mucho mas simplificada el logro de un determinado objetivo. Por eso es muy bueno que usted tenga algún leve conocimiento de los tipos de datos que permite el lenguaje de programación con el cual va a codificar sus algoritmos.

No olvide tampoco que siempre que usted almacene un valor determinado en una variable, el valor anterior se pierde pues el nuevo valor reemplaza el dato anterior. Por eso cuando usted necesite almacenar el dato anterior de una variables pues sencillamente tendrá que utilizar otra para que ese dato no se el pierda. Igualmente en algunos lenguajes es posible almacenar en una variable de un tipo de dato otro tipo de dato, esto quiere decir que en algunos lenguajes puede usted

almacenar un entero en una variable de tipo carácter. Claro que recalco que esta es una particularidad de determinados lenguajes de programación. Por estas razones es muy importante que usted conozca algunas características del lenguaje de programación con el cual va a codificar sus algoritmos ya que podrá utilizar apropiadamente algunos recursos de dicho lenguaje en la construcción de sus algoritmos.

También es muy importante que tenga en cuenta que por las características técnicas de los tipos de datos en los lenguajes de programación los datos de tipo entero y real tienen unos topes de almacenamiento. Para ello es muy bueno usted conozca esos topes de almacenamiento y si se da el caso del siguiente enunciado *Leer un número entero y determinar si tiene mas de 40 dígitos* usted inmediatamente sepa que con los tipos de datos convencionales no se puede desarrollar.
En cuanto a la precedencia de ejecución de los operadores tenga en cuenta que los que primero se resuelven son los paréntesis mas internos o sea aquellos paréntesis que no tienen mas paréntesis adentro. Esto es aceptado por todos los lenguajes de programación. También es importante que sepa que primero se desarrollan las potencias, luego las multiplicaciones y divisiones y luego las sumas y restas. Esta precedencia tanto con paréntesis como con operadores es lo que permite que una misma operación por compleja que sea tendrá exactamente los mismos resultados en cualquier lenguaje de programación dado que se rigen por las mismas reglas.

Cuando, al desarrollar una prueba de escritorio, tenga que resolver una expresión que incluya paréntesis y operadores y obviamente variables, no vaya a suponer nada, resuelva la operación tal y como lo haría el computador, de esta manera y solo así usted podrá notar si los resultados son los correctos o no, es decir, si la expresión sirve para lograr el objetivo o no.

Acerca de las Estructuras Básicas

En el tiempo que me he desempeñado como programador y como profesor de Lógica de Programación he notado que realmente la solución de ningún problema se sale de estas tres estructuras. Sé que es un poco arriesgado asegurarlo pero también sé que este libro busca exponer, como su nombre lo indica, la esencia de la lógica de programación y puedo garantizarle que el acertado manejo de estas tres estructuras es realmente la esencia en mención.

La estructura de secuencias no es mas que la aceptación del principio mínimo de trabajo de los algoritmos es decir el hecho de que cada orden se ejecuta después de la anterior y antes de la posterior. Además que todas las órdenes se van a ejecutar secuencialmente es decir desde el principio hasta el fin. Podríamos decir que esta estructura es la que gobierna todo el concepto general de la programación y a la vez permite mantener el hilo lógico conceptual del diseño de algoritmos.

La segunda estructura está formada por las decisiones. Ya hemos definido que no es mas que la escogencia de uno de entre dos ramales lógicos que dependen de una condición. Una decisión está formada por una formulación de una condición, si ésta es Verdadera entonces se ejecutarán las órdenes correspondientes si no lo es entonces se ejecutarán las otras órdenes. Es muy importante que usted sepa que las órdenes a ejecutar en caso de que la condición sea Verdadera

estarán escritas entre la formulación de la condición y la palabra clave *Sino* (en caso de que se considere la parte falsa de la condición) ó la palabra *Fin_Si* (que finaliza la decisión).

Las órdenes a ejecutar en el caso de que la condición sea Falsa estarán comprendidas entre la palabra clave *Sino* y el fin de la condición o sea el *Fin_Si* correspondiente. Para ello tenga en cuenta que el planteamiento de una condición no necesariamente involucra un *Sino* correspondiente ya que existen decisiones en donde si su condición no se cumple no nos interesa ejecutar alguna orden determinada.

Utilice apropiadamente los operadores relacionales y los operadores booleanos, para ello recuerde que un operador relacional permite obtener una respuesta de Verdadero o Falso y un operador Booleano permite conectar expresiones relacionales. La notación tanto para operadores relacionales como para operadores booleanos no es standard razón por la cual usted deberá saber cómo se expresa cada en el lenguaje de programación con el cual va a codificar sus algoritmos. Incluso puede utilizar esa notación en el desarrollo de sus algoritmos ya que de una vez se va acostumbrando a la notación propia del lenguaje de programación que va a utilizar.

Algunos lenguajes también permiten realizar operaciones de asignación en medio de una expresión lógica y por lo tanto mas de una vez van a facilitar el logro de un determinado objetivo. Como usted puede ver aunque no se desmiente que el algoritmo es el soporte lógico para lograr un objetivo tampoco se puede negar la inmensa importancia del conocimiento del lenguaje de programación con el cual se van a convertir en programas lo que en el momento solo sean algoritmos expresados en seudocódigo. Normalmente cada lenguaje cuenta con recursos y elementos que van a facilitar la construcción de unos algoritmos y por lo tanto el logro de unos objetivos, aunque dificulten un poco el logro de otros. Eso es lo que justifica precisamente la existencia de tantos lenguajes de programación en el medio informático.

Como se pudo ver, para la toma de decisiones también se cuenta con una estructura que nos permite escoger uno de entre varios ramales lógicos (me refiero a la estructura Casos). Cada lenguaje de programación tiene una implementación propia de esta estructura de decisión pero en esencia la lógica es la misma o sea que vamos a encontrar una equivalencia entre lo que nosotros programemos y la correspondiente codificación. Siempre tenga en cuenta que para que el computador tome una decisión, ésta tendrá que expresarse en términos de variables, constantes y operadores (tanto relacionales como booleanos).

Procure revisar bien sus algoritmos para que verifique si todas las decisiones son absolutamente necesarias dado que una de las órdenes que mas le toma tiempo al computador es tomar una decisión. Obviamente que hablo en términos del mismo computador pues el trabaja en millonésimas de segundo y por lo tanto lo que para nosotros pudiera parecer muy veloz para él pudiera parecer muy lento. Si un computador demora un segundo en resolver una decisión estará demorando demasiado ya que en ese mismo lapso de tiempo podrá realizar millones de operaciones. Por este motivo es muy importante que usted tenga en cuenta que uno de los factores que permiten medir la eficiencia de un programa es la cantidad de decisiones que el mismo involucre.

Muchas veces colocamos decisiones en nuestros algoritmos, a diestra y siniestra, desconociendo que este factor va en detrimento de la misma solución lógica del algoritmo. Por lo tanto siempre que usted haya terminado de desarrollar un algoritmo y lo haya probado y vea que cumple plenamente con el objetivo, tómese un tiempito para revisar cuáles decisiones pueden estar convirtiendo su solución en algoritmo ineficiente lo cual no contradice el logro del objetivo pues

usted puede llegar hasta la plaza de su pueblo recorriendo todas las calles o recorriendo solo aquellas que lo llevan allá en línea recta.

Los ciclos son la tercera estructura y con ellos podemos, normalmente, lograr desarrollar algoritmos mucho mas genéricos, es decir, sin las restricciones que sí se necesitan cuando no se tiene esta estructura a disposición. Los ciclos no son mas que esquemas que nos permiten, en cualquier lenguaje de programación, repetir conjuntos de instrucciones una cantidad finita de veces de tal forma que, acorde con un manejo correcto de variables, se pueda lograr mas de un objetivo de manera absolutamente genérica y sin tener que estar atado a condiciones especiales.

Con los ciclos todo lo que tenemos que hacer es tener mucho cuidado de que las condiciones sean coherentes con el cuerpo de los mismos de manera que se cumpla que el conjunto de instrucciones a repetir se itere una cantidad definida de pasos. Es muy útil recordar que la buena utilización de operadores relacionales y operadores booleanos es lo que nos permite lograr unos ciclos eficientes y ante todo muy funcionales.

Acerca de las Técnicas de Representación de Algoritmos

Hemos revisado los conceptos acerca de las diferentes técnicas de Representación de algoritmos y hemos mostrado cómo quedarían algunos de ellos representados con dichas técnicas. Las técnicas lo que van a permitir, cada una en su estilo, es colocar a su disposición herramientas para que la representación se haga mucho mas sencilla porque si usted se detiene a pensar un momento no es tan fácil representar una idea por su misma naturaleza etérea e intangible. Por este motivo la explicación de estas técnicas buscan que usted pueda tener unos elementos conceptuales que le permitan de una manera sencilla y simplificada representar sus ideas.

La diagramación libre permitió durante mucho tiempo que la idea general de solución de un problema fuera expresada de manera gráfica. Así se pudo, poco a poco, ir estandarizado soluciones para determinados problemas comunes. Como se pudo ver se explicó un cuadro comparativo de las diferentes técnicas y allí se expusieron las ventajas y desventajas de la diagramación libre. Considero que la diagramación como tal tiene la ventaja de mostrar lo que podríamos decir que es un “dibujo” que sirve como solución a un problema determinado o sea que sirve para lograr un objetivo planteado.

Aún a pesar de que realizar correcciones a un diagrama de flujo ya realizado es una tarea en mas de una ocasión dispendiosa e incómoda, aún así podría asegurarles que uno entiende cualquier cosa de una manera mucho mas fácil si se la explican a través de un gráfico, por esta razón es mucho mas fácil entender un mapa que entender una dirección. Este es el factor realmente rescatable de la técnica de diagramas de flujo sin embargo el gran problema que se presentaba al momento de realizarle una o mas correcciones es lo que ha desplazado poco a poco la utilidad de esta técnica. Además que resultaba ser muy desventajoso, hablando en términos técnicos, el hecho de que el programador tenga toda la libertad de hacer lo que le venga en gana en el papel para representar sus ideas dado que si bien algunos programadores son muy ordenados otros no lo son y si después le corresponde a alguien codificar lo que en su momento un programador altamente desordenado planteó a través de una maraña de líneas y símbolos que él erróneamente llamó diagrama de flujo se verá en grandes aprietos para entenderlo.

Por este motivo poco a poco se fue viendo la necesidad de buscar otra forma de expresión de los algoritmos. Fue allí en donde nació la idea de la Diagramación Rectangular Estructurada que permitía que el programador expresara cualquier idea de una manera aproximadamente gráfica pero no con tanta libertad como se hacía en los diagramas de flujo. También presentaba la pequeña dificultad de realizar correcciones pero se hacía de todas formas mucho mas fácil la tarea de corregir que en los diagramas de flujo. No se puede negar que la representación gráfica como tal de un algoritmo a través de la Diagramación Rectangular Estructurada no es tan entendible a primera vista como sí lo es con los Diagramas de Flujo pero si se debe aceptar que es mucho mas técnica y por ello facilita mucho la posterior codificación, otro gran problema que colocaban los diagramas de flujo.

Bajo mi óptica diría que a alguien se le ocurrió quitarle las líneas a la Diagramación Rectangular Estructurada y adicionarle unos nuevos estandares y prácticamente fue así como se llegó al concepto del seudocódigo que pasó a ser la solución óptima dentro del objetivo de lograr una forma de representación de ideas que permitiera fácilmente una posterior codificación, que tuviera unas reglas establecidas por la misma lógica y que además fuera, desde todo punto de vista, entendible.

El seudocódigo abre un camino para una posterior codificación simplificada y además es muy entendible, teniendo en cuenta que se circunscribe a unas reglas lógicas que estandarizan el desarrollo del algoritmo como tal. Sin desconocer las inmensas bondades de las otras dos técnicas, considero que el seudocódigo permite lograr de una manera muy sencilla la representación de una determinada idea solución orientada al logro de un determinado objetivo.

Finalmente es muy bueno que usted conozca las tres técnicas y de vez en cuando las practique dado que existen muchos autores de manuales y de libros que utilizan indistintamente una u otra y entonces allí se hará muy importante que las conozcamos para que podamos entender lo que nos quieren representar técnicamente.

Acerca de la Tecnología

Usted ha podido notar que en el mundo actual existe un despliegue de tecnología que no solo facilita sino que también condiciona la vida del ser humano. Los lenguajes de programación no son la excepción dentro de esa carrera comercial en la cual se encuentran inmersas todos fabricantes de tecnología. Con gran frecuencia se encuentra uno con la decepción de que el software de desarrollo o sea los lenguajes de programación que existen en el mercado son modificados y, teóricamente, mejorados con tanta vertiginosidad que uno escasamente alcanza a conocer algunas de sus bondades cuando ya las están cambian por otras.

Por eso no hay que preocuparse de que la tecnología va avanzando mucho mas rápido de cómo avanzan nuestros conocimientos pues lo que nos corresponde es tratar de estar a la vanguardia en dichos avances. Cómo se hace para estar al día con todos los lenguajes de programación..? Pues si usted en algún momento ha tenido este propósito es verdaderamente imposible.

Permanentemente trate de mantenerse actualizado en cuanto a los avances del Lenguaje de Programación con el que usted usualmente trabaja. Si está vinculado a una empresa en donde se trabaja con un lenguaje determinado pues entonces ha de procurar aprovechar las políticas de actualización de la empresa para tratar de mantenerse al día en las mejoras que dicho lenguaje vaya teniendo.

Sin embargo tenga en cuenta que por mas que se estudie es muy posible que usted siempre esté un poco rezagado de las actualizaciones de los lenguajes de programación dado que esta tecnología nos llega luego de que ha pasado algún tiempo de prueba y otro de comercialización en los países industrializados. La ventaja de trabajar con un solo lenguaje de programación es que de una u otra forma usted estará, por la simple necesidad de uso del mismo, mucho mas actualizado que lo normal (obviamente si la empresa tiene unas políticas de actualización tecnológica apropiadas). La desventaja es que usted se va a ir "encasillando" poco a poco en un solo lenguaje y probablemente ni siquiera llegue a conocer las ventajas y los avances de otros lenguajes de programación, sin embargo ha de entender y aceptar usted que no es fácil para una empresa estar cambiando su plataforma de desarrollo cada seis meses solo porque salió un lenguaje de programación aparentemente mejor.

Por esta razón se hace muy importante ir poco a poco conociendo las actualizaciones en determinado lenguaje y, sin "casarse" de manera exclusiva con él, si tener alguna preferencia por un lenguaje que por los demás. Considero que eso le permite a uno tener un poco mas de destreza en la programación así sea orientando sus algoritmos a la implementación en un lenguaje determinado. Tenga en cuenta que cada vez que usted se sienta a desarrollar un algoritmo, ha de buscar que dicho algoritmo sea lo mas genérico posible no solo en su concepción lógica sino en su posterior implementación, es decir, el algoritmo debe estar escrito en términos que se facilite la codificación en cualquier lenguaje. No puedo negarle que en la medida en que uno se acostumbra a un determinado lenguaje de programación en esa misma medida se va acostumbrando a la utilización de elementos y conceptos propios de dicho lenguaje en la construcción de los algoritmos. Bueno o malo..? Eso no sabría respondérselo pues encuentro tantas cosas buenas como malas pero sé que es así como medianamente podemos llegar a funcionar teniendo en cuenta que definitivamente los lenguajes de programación avanzan mucho mas rápido que nuestro aprendizaje o al menos así sucede en el mercado informático colombiano.

Considero que la mejor fuente para tener una percepción de los avances de los lenguajes de programación así dicha percepción solo sean ideas de mejoramiento mas no instrucciones puntuales en sí, la constituyen las revistas técnicas en donde explican bondades y beneficios de las mejoras de los lenguajes de programación. Estas revistas lo que permiten es que tengamos una idea del camino que están siguiendo los avances en el desarrollo de las herramientas de programación. Ya si nos interesa conocer en detalle dichas mejoras entonces tendríamos que trabajar con el lenguaje en mención y de esta forma podríamos conocer, de manera aplicativa, cuales han sido dichos avances.

Utilice preferiblemente uno o dos lenguajes de programación si es independiente. Si la utilización de dicho lenguaje de programación está determinada por la empresa entonces estudie todas sus bondades. Estoy seguro que mas de una vez se encontrar con muchas instrucciones del mismo lenguajes desconocidas para usted y que le sacarán de mas de un lío programático.

Acerca de las Decisiones

En cuanto a las decisiones realmente es poco lo que hay que acotar dado que en la sección Acerca de las Estructuras Básicas se ha dicho la mayoría. Tal vez podría agregar que no se ate solo a la estructura Si-Entonces-Sino pues también tenga en cuenta que la estructura casos le va a permitir simplificar mas de un conjunto de decisiones y hacer que su algoritmo sea mucho mas entendible y fácil de codificar.

Siempre evalúe bien las condiciones de las decisiones y realícele la prueba de escritorio a las decisiones tal y como se ha recomendado en todo este libro, es decir, sin suponer absolutamente nada y sin saltarse ningún paso debido a que se puede encontrar, mas de una vez, con que saltarse una instrucción que involucre una decisión va a significar un cambio realmente profundo en el desarrollo del algoritmo y en su posterior ejecución. Revise que las decisiones que aparezcan en su algoritmo realmente sean necesarias y omita todas aquellas que, por razones estrictamente lógicas, estén sobrando.

Las decisiones son el único camino que tenemos en la programación para escoger uno de entre dos ramales lógicos por eso es una de las herramientas que debemos utilizar de manera eficiente pues su excesivo uso, tal como ya se expuso, representa ineficiencia en la construcción del algoritmo. Entre menos decisiones tenga un algoritmo mas eficiente será la ejecución de su correspondiente programa.

Acerca de los Ciclos

Como vimos existen diferentes tipos de ciclos y la mayoría de ellos están contemplados en los lenguajes de programación. Normalmente me preguntan cuál de las estructuras cíclicas es la mejor..? A lo cual yo respondo que realmente esta pregunta no se puede resolver en cuanto a una comparación y escogencia del mejor ciclo. Comencemos por decir que en la inmensa mayoría de las veces todos los ciclos son equivalentes. Como vimos existían unos casos que no permitían una implementación de algunos ciclos pero en general la mayoría de las veces son equivalentes.

No puedo negarle que uno poco a poco se va acostumbrando a la permanente utilización de uno o dos esquemas de ciclos pero ello no es porque sean mejores o no, sencillamente es porque uno se acostumbra a ellos y no mas. Cuando necesite un ciclo en un algoritmo simplemente utilice aquel que usted vea mas a la mano, es decir, aquel que vea mas lógico, aquel con el cual usted vea que le resulta mas fácil implementar el conjunto de instrucciones a repetir. No se ate obligatoriamente a un esquema de ciclos pero no se extrañe si con el tiempo nota que utiliza uno o dos esquemas de ciclos muchos mas que los otros.

La forma de escritura de algunos esquemas de ciclos en determinados lenguajes de programación puede llegar a ser un poco confusa y en muchos casos casi caprichosa. Tenemos que acostumbrarnos a ello si estamos trabajando con un lenguaje de estos o si nos toca trabajar con ellos. La sintaxis (que no es mas que la forma como se escriben las órdenes en un lenguaje de

programación) está diseñada por los creadores del lenguaje y por lo tanto no es cambiable a nuestro gusto aunque puedo garantizarle que no se preocupe que si usted se acostumbra a (o le toca) trabajar con un determinado lenguaje de programación, por mas difícil que sea usted en pocos días verá que su sintaxis es entendible y hasta llegará a defenderla al fin y al cabo somos unos “animales de costumbre”.

Acerca de los Vectores

Los vectores establecen una herramienta de programación tan fuerte que mas de una vez vamos a recurrir a ellos para que nuestros algoritmos sean realmente eficientes. No sobredimensione los vectores, esto quiere decir que no utilice vectores con una dimensión exagerada para no correr el riesgo de quedarse corto de memoria. Trate de ser lo mas exacto en la dimensión de los vectores pues tenga en cuenta que sobredimensionarlos implica estar reservando espacios de memoria que probablemente nunca se van a utilizar en la práctica.

Sé que usted estará pensando que si no existe una técnica que permita darle a los vectores una dimensión dependiendo de las necesidades instantáneas de ejecución del programa. La verdad si existe una técnica y no es exactamente con vectores pero no son tema de este libro sin embargo en la medida en que usted utilice bien los vectores empezará a notar como sus programas no solo pueden llegar a aprovechar eficientemente los recursos del computador sino que logrará objetivos cada vez mas genéricos y con menos esfuerzo tecnológico.

Siempre que utilice un vector tenga en cuenta la buena utilización de una variable que servirá como subíndice para referenciar cada uno de los elementos almacenados en el vector. No olvide que cuando se hace referencia a *V [i]* se está citando el valor que se encuentra almacenado en el vector V en la posición *i* y cuando se hace referencia a *i* sencillamente se está citando la posición de un determinado dato del vector. Este ejemplo es válido si asumimos que estamos dentro de un algoritmo en donde estamos utilizando un vector que hemos llamado *V* y que utiliza un subíndice que hemos llamado *i.* De la buena utilización de subíndices en los vectores depende la buena aplicación de los mismos en los algoritmos.

Acerca de las Matrices

Las matrices al igual que los vectores tienen una precaución en su utilización y es que no se vayan a sobredimensionar por el temor de quedarnos cortos de memoria. Siempre trate al máximo de que la dimensión de las matrices que utilice sea lo mas justa posible precisamente para que usted no esté reservando todo un espacio que posteriormente dentro del algoritmo no va a utilizar. Trate al igual que con las variables y los vectores de que los nombres de las matrices sean lo mas mnemónicos posibles pues de esto también dependerá que en cualquier momento usted vea un algoritmo y se ubique en cuanto a su concepción lógica.

Tenga también en cuenta que la utilización de matrices normalmente exigirá la utilización de dos subíndices, uno para las filas y otro para las columnas, y que no necesariamente estor dos subíndices tienen que llamarse i y j. Sencillamente necesitan ser dos variables enteras y nada mas. Al igual que con los vectores de la buena utilización de los subíndices de una matriz obtendremos la buena aplicación del concepto de matriz dentro de nuestros algoritmos. No utilice matrices en donde no las vaya a necesitar, siempre verifique si realmente el enunciado exige la presencia de matrices o no para que no tenga que estar reservando espacios de memoria que no va a usar.

Si los datos que usted lee deben ser almacenados, por alguna razón, en forma de filas y columnas entonces usted necesitará una matriz. Si necesita que además de la consideración anterior, sus datos permanezcan para una posterior utilización entonces usted también necesitará en ese momento la utilización de una matriz. Siempre encontrará razones lógicas que van a validar o a rechazar la utilización de matrices dentro de la estructura lógica de un algoritmo.

Acerca de las Funciones

Las funciones son, como yo les llamo en todos los cursos de programación que he dictado, la gran "vedette" de la programación pues con las funciones como pudimos ver en el contexto del libro se solucionan los que considero que son los problemas mas grandes que tiene la programación y que precisamente no son técnicos sino de concepción humana. Cuando usted vaya a construir una función tenga en cuenta que entre mas genérica la construya mas le va a servir para mas adelante.

Cómo lograr funciones verdaderamente genéricas..? Verificando qué parámetros necesita. Solo allí usted podrá encontrar el verdadero sentido de las funciones y su gran utilidad. En mi concepto la programación conceptualmente no ha entregado una estructura mas práctica y mas simplificada que el concepto de función. Yo diría que hasta de allí en adelante la programación no ha evolucionado pues toda la teoría actual de la programación se basa en esa unidad de trabajo, en esa "célula" funciona. Por tal motivo yo le diría que si al leer usted este libro y llegar a este punto, encuentra que no solo ha entendido claramente que es una variables, qué es una secuencia, qué es una decisión, qué es un ciclo, qué es un vector y qué es una matriz y además ha entendido claramente qué es una función y como se aplican en ella todos los conceptos anteriores entonces este libro logró su objetivo inicial que era colocar a su disposición todos los elementos básicos que constituyen la esencia de la lógica de programación.