Indice

tecnicas de diseño de programas

El proceso de diseño comprende al desarrollo de una visión conceptual del sistema, el establecimiento de una estructura, la identificación de las cadenas de datos y su almacenamiento, la descomposición de funciones de alto nivel en su subfunciones, el establecimiento de las relaciones e interconexiones entre componentes, el desarrollo de la representación de datos en forma concreta y la especificación de los detalles de los algoritmos.

Las técnicas de diseño comúnmente están basadas en las estrategias de las jerarquías de `'hacia abajo" y de `'hacia arriba" . Por medio del enfoque de arriba a abajo, se pone atención inicialmente en los aspectos globales de todo el sistema ; conforme el diseño progresa, el sistema se descompone en subsistema, poniéndosele el mayor consideración a los detalles específicos. El encadenamiento hacia atrás resulta fundamental en este tipo de diseño. Con el fin de reducir este encadenamiento hacia atrás, muchos diseñadores proponen el uso de una estrategia mezclada, la cual es predominantemente hacia abajo, pero que primero requiere de la especificación de los módulos inferiores. La ventaja primordial de esta estrategia es que se dedica a la atención a las necesidades del cliente, a las interfaces con el usuario y a la naturaleza global del problema a resolver.

En el enfoque hacia arriba del diseño de productos de programación, el diseñador primero intenta identificar al conjunto primitivo de objetos acciones y relaciones que proporcionarán una base para la solución del problema ; los conceptos de alto nivel son después formulados en términos del conjunto de primitivos. La estrategia hacia arriba requiere que el diseñador combine las características proporcionadas por el lenguaje de instrumentación para dar entidades son a su vez combinadas hasta que se construye un conjunto de funciones, estructuras de datos e interconexiones para resolver el problema por medio del uso de las facilidades del ambiente de programación existente ; este tipo de diseño puede también requerir del rediseño y el encadenamiento hacia atrás del mismo. El éxito de este enfoque depende de la identificación del conjunta adecuado de ideas primitivas que sean suficientes para la instrumentación del sistema.

Top down

El diseño descendente es una técnica que permite diseñar la solución de un problema con base en la modularización o segmentación dándole un enfoque de arriba hacia abajo (Top Down Design). Esta solución se divide en módulos que se estructuran e integran jerárquicamente, como si fuera el organigrama de una empresa. Ejemplo :

ALGORITMO

ALGO

MODULO MODULO MODULO

UNO DOS TRES

En el diagrama anterior se muestra la estructura del algoritmo ALGO, que se auxilia de tres módulos subordinados, cada uno de los cuales ejecuta una tarea especifica. En su momento el modulo principal ALGO invocara o llamara a los módulos subordinados, es decir, dirigirá su funcionamiento.

¿ Que es un modulo ?

Un modulo es un segmento, rutina, subrutina, subalgoritmo o procedimiento, que puede definirse dentro de un algoritmo con el fin de ejecutar una tarea especifica y puede ser llamado o invocado desde el algoritmo principal cuando sea necesario.

¿ Cuando es útil la modularización ?

Este enfoque de segmentación o modularizacion es útil en dos casos :

Cuando existe un grupo de instrucciones o una tarea especifica que deba ejecutarse en más de una ocasión.

Cuando un problema es complejo o extenso, la solución se divide o segmenta en módulos que ejecutan partes o tareas especificas . Dicha solución se organiza de forma similar a como lo hacen las empresas cuando se estructuran con base en las funciones para realizar sus actividades ; en otras palabras, el trabajo se divide en partes que sean fácilmente manejables y que , lógicamente, puedan ser separadas ; así, cada una de estas partes se dedica a ejecutar una determinada tarea, lo que redundara en una mayor concentración , entendimiento y capacidad de solución a la hora de diseñar la lógica de cada una de estas. Dichas partes son módulos o segmentos del algoritmo, algunos de ellos son los módulos directivos o de control, que son los que se encargaran de distribuir el trabajo de los demás módulos. De esta manera se puede diseñar un organigrama que indique la estructura general de un algoritmo.

En el diagrama anterior se tiene un modulo directivo llamado algoritmo ALGO, que dirige el funcionamiento de tres módulos subordinados, que son : MODULO UNO, MODULO DOS y MODULO TRES .

PROCESO DE MODULARIZACION.

El proceso de segmentación consiste en hacer una abstracción del problema, del cual se tiene inicialmente un panorama general. Enseguida, se procede a desmenuzar o dividir el problema en partes pequeñas y simples, como se muestra :

Se forma un primer modulo enunciando el problema en términos de la solución a éste.

EJEMPLO :

Se necesita diseñar un algoritmo que ayude a un niño a revisar sus tareas referentes a las operaciones aritméticas fundamentales : sumar, restar, multiplicar y dividir. El proceso es el siguiente :

Se ofrecerá un menú de opciones para escoger lo que desee hacer, de acuerdo con el siguiente formato :

TE PUEDO AYUDAR A :

SUMAR

RESTAR

MULTIPLICAR

DIVIR

FIN

OPCION

En el caso de la suma el procedimiento es el siguiente : leer los dos números y el resultado obtenido por el niño, hacer el calculo de la maquina, comparar ambos resultados y decirle si esta correcto o incorrecto. Se le puede permitir realizar las revisiones de operaciones de suma que sean necesarias. Para el caso de la resta, multiplicación y división se seguirá un procedimiento similar, claro está, con las diferencias existentes.

APLICACIÓN : Aplicar lo enunciado en este punto y si se considera que se trata de un algoritmo que ayuda a un niño en la revisión de sus tareas, se tiene el modulo principal siguiente :

ALGORITMO

AYUDA

Se toma este modulo y se busca la forma de dividirlo en otros módulos más pequeños, que ejecuten tareas o funciones especificas. Las mismas funciones que se desea que ejecute el algoritmo, nos darán la pauta para definir los módulos, y así hacer una segmentación de la solución del problema en partes manejables.

APLICACIÓN :

Si nos referimos al ejemplo anterior, tenemos que se deben mantener cuatro tareas o funciones claramente definidas : sumar, restar, multiplicar y dividir, de ahí que necesitamos un modulo para cada una de las tareas, las cuales deberán estar subordinadas al modulo principal. La estructura que se tiene, entonces, es la siguiente :

ALGORITMO

AYUDA 1

MODULO MODULO MODULO MODULO

2 3 4 5

SUMAR RESTAR MULTIPLICAR DIVIDIR

NOTA : Los módulos se numeran de arriba hacia abajo y de izquierda a derecha.

3. Se repite el paso 2 para cada módulo nuevo definido, hasta llegar a un nivel de detalle adecuado, es decir, hasta hacer que cada modulo ejecute una tarea especifica, que este claramente definida y que el diseño de la lógica del mismo resulte fácil utilizando el pseudocódigo.

APLICACIÓN :

En el problema que estamos analizando, se revisan los sumar, restar, multiplicar y dividir. Se considera que

estos tienen un nivel de detalle adecuado, porque cada modulo hace una tarea muy simple, clara y especifica y, en consecuencia, se pueden diseñar fácilmente utilizando el pseudocódigo.

En caso de no ser así, es decir, que el módulo restar, por ejemplo, requiera otros módulos subordinados, la escritura general del algoritmo sería la siguiente :

ALGORITMO

1

AYUDA

MODULO MODULO MODULO MODULO

2 3 7 8

SUMAR RESTAR MULTIPLICAR DIVIDIR

MODULO MODULO MODULO

4 5 6

R1 R2 R3

Como se puede ver la numeración sufre ciertas alteraciones, es decir, cuando se llega al modulo restar se enumera todo lo que dependa de este, para después continuar con el de multiplicar que es el siguiente en el orden.

En este caso, el modulo de restar se convierte a su vez en un modulo directivo que controla la ejecución de sus módulos subordinados.

FORMA DE UTILIZAR EL DISEÑO DESCENDENTE CON PSEUDOCÓDIGO :

Cuando el enfoque de diseño descendente se utiliza de manera conjunta con la técnica del pseudocódigo, se logra una herramienta de diseño de algoritmos muy poderosa.

Procedimiento para usar dicha herramienta :

Se tiene el análisis de un problema.

Se diseña la estructura general del algoritmo utilizando el diseño descendente.

Se toma cada modulo y se diseña su lógica utilizando el pseudocódigo.

COMO LLAMAR MODULOS EN PSEUDOCODIGO :

Llamar NOM_MODULO

DONDE :

Llamar identifica la acción de llamar o invocar a los módulos.

NOM_MODULO es el nombre del modulo que se esta llamando.

EJEMPLO :

A continuación se presenta el diseño de la solución completa del problema relacionado con el diseño del algoritmo que ayude a un niño en la revisión de sus tareas de operaciones aritméticas fundamentales.

La estructura del algoritmo es :

ALGORITMO

1

AYUDA

MODULO MODULO MODULO MODULO

2 3 4 5

SUMA RESTA MULTIPLICACION DIVISION

El siguiente paso es diseñar la lógica de cada modulo utilizando el pseudocódigo.

Algoritmo AYUDA

Definir variables

N1,N2, RESNI, RESMA : Entero

DESEA OTRO : Alfabético [1]

OPCION : Entero

REPEAT

Imprimir el menú de opciones

TE PUEDO AYUDAR A :

SUMAR

RESTAR

MULTIPICAR

DIVIDIR

FIN

ESCOGER OPCION :

Leer OPCION

CASE (1,2,3,4) OPCION

1 : Llamar SUMAR

2 : Llamar RESTAR

3 : Llamar MULTIPLICAR

4 : Llamar DIVIDIR

ENDCASE

UNTIL OPCION= 5

Fin

Modulo SUMAR

REPEAT

Solicitar numero uno, numero dos y resultado

Leer N 1, N2, RESNI

Calcular RESMA= N 1 + N2

IF RESMA = RESNI THEN

Imprimir `correcto'

ELSE

Preguntar `¿Desea seguir intentando (S/N) ?'

Leer OTRO

DOWHILE OTRO= ?S ?

1. Leer RESNI

IF RESNI = RESMA THEN

Imprimir `correcto'

OTRO= `N'

ELSE

Imprimir `incorrecto'

Preguntar `¿Desea seguir (S/N) ?'

Leer otro

ENDIF

ENDO

ENDIF

Solicitar `¿Desea sumar de nuevo(S/N) ?'

Leer DESEA

UNTIL DESEA = `N `

Fin modulo SUMAR

Modulo RESTAR

REPEAT

Solicitar número uno, numero dos y resultados

Leer N1,N2,RESNI

Calcular RESMA = N1−N2

IF RESMA= RESNI THEN

imprimir `correcto'

ELSE

imprimir `incorrecto'

Preguntar `¿Desea seguir intentando (S/N) ?'

Leer OTRO

DOWHILE OTRO='S'

Leer RESNI

IF RESNI = RESMA THEN

Imprimir `correcto'

OTRO='N'

ELSE

Imprimir ` incorrecto `

Preguntar `¿Desea seguir (S/N) ?'

Leer OTRO

ENDIF

ENDDO

ENDIF

Preguntar `¿Desea restar de nuevo (S/N) ?'

Leer DESEA

UNTIL DESEA='N'

Fin modulo RESTAR

Modulo MULTIPLICAR

REPEAT

Solicitar numero uno, numero dos y resultado

Leer N1,N2,RESNI

Calcular RESMA=N1*N2

IF RESMA=RESNI THEN

Imprimir `correcto'

ELSE

Imprimir `incorrecto'

Preguntar `¿desea seguir intentando (S/N) ?'

Leer OTRO

DOWHILE OTRO='S'

Leer RESNI

IF RESNI = RESMA THEN

Imprimir `correcto'

OTRO='N'

ELSE

Imprimir `incorrecto

Solicitar `¿Desea seguir intentando (S/N) ?'

Leer OTRO

ENDIF

ENDDO

ENDIF

Solicitar `¿Desea multiplicar de nuevo (S/N) ?'

Leer DESEA

UNTIL DESEA= `N'

Fin modulo MULTIPLICAR

Modulo DIVIDIR

REPEAT

Solicitar numero uno, numero dos y resultado

Leer N1, N2, RESNI

Calcular RESMA=N1/N2

IF RESMA=RESNI THEN

Imprimir `correcto'

ELSE

Imprimir `incorrecto'

Preguntar `¿Desea seguir intentando (S/N) ?'

Leer OTRO

DOWHILE OTRO='S'

Leer RESNI

IF RESMI=RESMA THEN

Imprimir `correcto'

OTRO='N'

ELSE

Imprimir `incorrecto'

Preguntar `¿Desea seguir intentando (S/N) ?'

Leer OTRO

ENDIF

ENDDO

ENDIF

Solicitar `¿Desea dividir de nuevo (S/N) ?'

Leer DESEA

UNTIL DESEA='N'

Fin modulo DIVIDIR

FUNCIONES

La función es una estructura autónoma similar a los módulos. La diferencia radica en que la función se relaciona especificando su nombre en una expresión, como si fuera una variable ordinaria de tipo simple. Las funciones se dividen en estándares y definidas por el usuario.

FUNCIONES ESTANDAR

Son funciones proporcionadas por cualquier lenguaje de programación de alto nivel, y se dividen en aritméticas y alfabéticas.

ARITMETICAS : seno, coseno, Arctan(arco tangente), Ln (logaritmo natural),Exp (e elevada a la x potencia),Absoluto,

redondeo, cuadrado, raiz, truncar, fracción.

ALFABETICAS : carácter, ordinal, concatenar,borrar,insertar, longitud, posicion, NUMCAD(convierte el valor numérico real o entero en cadena),CADNUM(convierte la cadena en un numero real o entero).

FUNCIONES DEFINIDAS POR EL USUARIO

Son funciones que puede definirlas el programador con el propósito de ejecutar alguna función especifica, y que por lo general se usan cuando se trata de hacer algún calculo que será requerido en varias ocasiones en la parte principal del algoritmo.

El nombre de la función puede estar seguido de uno o mas para metros actuales encerrados entre paréntesis. Por lo general transfieren datos a parámetros tipo valor.

Bottom up

La diferencia del tipo de diseño ascendente y descendente solo se puede dar a la hora de la programación. Porque en el momento de dibujar la estructura del problema, en las dos formas el diseño queda igual, solamente que los módulos son enumerados en forma diferente, pero esto se hace pensando ya en como se va a comenzar a programar. En el diseño ascendente primero se programan los módulos que se encuentran mas abajo de la estructura, hasta llegar al primer modulo dibujado.

Tomando un ejemplo del diseño descendente la estructura quedaría como sigue :

Ejercicio : Realiza el diseño ascendente para la gestión del control de un hotel utilizando la siguiente información :

- pagos a empleados (nombre, puesto, sueldo, horas extra)
- prestamos externos (cliente, préstamo, aval, plazo)
- libro de reservaciones(nombre, departamento, entrada, salida)
- mantenimiento (área, daños, total)

Esta gráfica muestra los módulos generales que va a contener el programa.

SUBPROGRAMA

CONTROL 20

SUBPROGRAMA SUBPROGRAMA SUBPROGRAMA SUBPROGRAMA

PAGOS A PRESTAMOS LIBROS DE MANTENIMIENTO

EMPLEADOS 5 EXTERNOS RESERVACIONES 15 19

WARNIER ORR

Los diagramas de Warnier/Orr (también conocidos como construcción lógica de programas/construcción lógica de sistemas) fueron desarrollados inicialmente en Francia por Jean Dominique Warnier y en los Estados Unidos por Kenneth Orr. Este método ayuda al diseño de estructuras de programas identificando la salida y resultado del procedimiento, y entonces trabaja hacia atrás para determinar los pasos y combinaciones de entrada necesarios para producirlos. Los sencillos métodos gráficos usados en los diagramas de Warnier/Orr hacen evidentes los niveles en un sistema y más claros los movimientos de los datos en dichos niveles.

ELEMENTOS BASICOS

Los diagramas de Warnier/Orr muestran los procesos y la secuencia en que se realizan. Cada proceso se define de una manera jerárquica ; es decir, consta de conjuntos de subprocesos que lo definen, en cada nivel, el proceso se muestra en una llave que agrupa a sus componentes. Puesto que un proceso puede tener muchos subprocesos distintos, un diagrama de Warnier/Orr usa un conjunto de llaves para mostrar cada nivel del sistema.

USO DE DIAGRAMAS DE WARNIER/ORR

la capacidad de mostrar la relación entre procesos y pasos de un proceso no es exclusiva de los diagramas de Warnier/Orr, así como tampoco lo es el uso de la iteración, selección de alternativas o el tratamiento de casos individuales. Tanto los diagramas de flujo estructurado y los métodos del español estructurado logran eso también. Sin embargo, el enfoque que se usa para desarrollar las definiciones de un sistema por medio de

estos diagramas es distinto y se adapta y se adaptan bien a los que se usan en el diseño de sistemas lógicos.

Para desarrollar un diagrama de Warnier/Orr , el analista trabaja hacia atrás, empezando con la salida del sistema y usando un análisis orientado hacia la salida. En el papel el desarrollo se mueve de izquierda a derecha. En primer lugar, se definen la salida o resultados esperados del procedimiento. En el nivel siguiente, mostrado mediante la inclusión por medio de una llave, se definen los pasos necesarios para producir la salida. A su vez, cada paso se define un poco mas. Las llaves adicionales agrupan los procesos requeridos para producir el resultado en el siguiente nivel.

Los diagramas de Warnier/Orr ofrecen a los expertos en sistemas algunas ventajas distintivas. Son simples en apariencia y fáciles de entender. Aun así, son poderosas herramientas de diseño. Tienen la ventaja de mostrar agrupaciones de procesos y los datos que deben transferirse de nivel a nivel. Además, la secuencia del trabajo hacia atrás garantiza que el sistema estará orientado hacia el resultado.

Conclusiones

En esta investigación abarcamos tres diferentes técnicas de diseño las cuales son el TOP DOWN, BOTTOM UP y el WARNIER/ORR.

La técnica TOP DOWN también es conocida como arriba−abajo y consiste en una serie de niveles de menor a mayor complejidad que dan solución al problema, en esencia consiste en efectuar una relación entre las etapas de la estructuración de forma que una etapa jerárquica y su inmediato inferior se relacionan mediante la entrada y salida de información.

La técnica BOTTOM UP es conocida también como ascendente, la diferencia entre el bottom up y el top down es que los módulos son enumerados de forma diferente. En el bottom up se enumeran primero los módulos inferiores hasta llegar al modulo superior.

El método Warnier/Orr se trata de un método para la representación de programas cuyo resultado final se llama diagrama de Warnier. En el podemos utilizar toda la terminología estudiada hasta ahora en lo que respecta a identificadores, constantes, variables, expresiones y operadores, teniendo en cuenta que la característica fundamental es la forma de diseñar el programa que será descendentemente y la representación utilizada.

20

5