GUÍA DE ESTRUCTURAS DE CONTROL

Hasta ahora nuestros algoritmos han consistido en simples secuencias de instrucciones unas depues de otra. Pero en nuestro programas existen tareas más complejas que no pueden ser resueltas así, quizás necesitamos repetir una misma instrucción, realizar acciones diferentes en función del valor de una expresión, etc. Para esto existen las estructuras de control. ESTRUCTURAS DE CONTROL Las Estructuras de Control determinan el orden en que deben ejecutarse las instrucciones de un algoritmo, es decir, si serán recorridas una después de la otra (estructuras secuenciales), si habrá que tomar decisiones sobre si ejecutar o no alguna acción (estructuras selectivas o de decisión) o si habrá que realizar repeticiones (estructuras repetitivas). Esto significa que una estructura de control permite que se realicen unas instrucciones y omitir otras, de acuerdo a la evaluación de una condición.

ESTRUCTURA SECUENCIAL

Es la estructura en donde una acción (instrucción) sigue a otra de manera secuencial. Las tareas se dan de tal forma que la salida de una es la entrada de la que sigue y así en lo sucesivo hasta cumplir con todo el proceso. Esta estructura de control es la más simple, permite que las instrucciones que la constituyen se ejecuten una tras otra en el orden en que se listan. ESTRUCTURAS SELECTIVAS Estas estructuras de control son de gran utilidad para cuando el algoritmo a desarrollar requiera una descripción más complicada que una lista sencilla de instrucciones. Este es el caso cuando existe un número de posibles alternativas que resultan de la evaluación de una determinada condición. Este tipo de estructuras son utilizadas para tomar decisiones lógicas, es por esto que también se denominan estructuras de decisión o selectivas. En estas estructuras, se realiza una evaluación de una condición y de acuerdo al resultado, el algoritmo realiza una determinada acción. Las condiciones son especificadas utilizando expresiones lógicas. 2 Las estructuras selectivas/alternativas pueden ser: • Simples: Si • Doble: Si- SiNo • Múltiples: Según – Si Anidado

CONDICIÓN SIMPLE

La estructura alternativa simple si-entonces lleva a cabo una acción siempre y cuando se cumpla una determinada condición. La selección si-entonces evalúa la condición y luego: • Si la condición es verdadera, ejecuta el bloque de acciones • Si la condición es falsa, no ejecuta nada. Pseudocódigo en PSeInt: Si expresión lógica Entonces acciones Fin Si Ejemplo para abrir en Pseint: Condicion Simple CONDICIÓN DOBLE La estructura anterior es muy limitada y normalmente se necesitará una estructura que permita elegir entre dos opciones o alternativas posibles, en función del cumplimiento o no de una determinada condición. Si la condición es verdadera, se ejecuta la acción S1 y, si es falsa, se ejecuta la acción S2. 3 Pseudocódigo en PSeInt: Si expresión lógica Entonces acciones_por_verdadero Sino acciones_por_falso Fin Si Ejemplo para abrir en Pseint: Condicion Doble CONDICIÓN MÚLTIPLE Muchas veces vamos a tener más de dos alternativas para elegir, o una variable que puede tomar varios valores. Para solucionar esto, usamos la condición multiple. En esta estructura, se evalúa una condición o expresión que puede tomar n valores. Según el valor que la expresión tenga en cada momento se ejecutan las acciones correspondientes al valor. La estructura de decisión múltiple evaluará una expresión que podrá tomar n valores distintos, 1, 2, 3, 4, ..., n. Según el valor que elija en la condición, se realizará una de las n acciones, o lo que es igual, el flujo del algoritmo seguirá un determinado camino entre los n posibles. Por ejemplo, si tenemos un sistema de notas, donde 6 es desaprobado, 7 es aprobado, 9 es sobresaliente y 10 es excelente. Al tener un valor que puede dar distintas alternativas, usamos la condición múltiple. Pseudocódigo en PSeInt: Segun variable_numerica Hacer opcion_1: secuencia_de_acciones_1 opcion_2: secuencia_de_acciones_2 opcion_3: secuencia_de_acciones_3 De Otro Modo:

secuencia_de_acciones_dom Fin Según NOTA: Cuando el valor de la variable que se evalúa no coincide con ninguno de los valores que se evalúa, entonces se ejecutan las acciones dentro del bloque “De Otro Modo” (secuencia_de_acciones_dom), el cual equivale a realizar un “Sino” dentro de las estructuras condicionales. 4 Este problema, se podría resolver por estructuras alternativas simples o dobles, anidadas o en cascada; sin embargo, este método si el número de alternativas es grande puede plantear serios problemas de escritura del algoritmo y naturalmente de legibilidad. Ejemplo para abrir en Pseint: Condicion Multiple

CONDICIONALES ANIDADOS O EN CASCADA Es posible también utilizar la instrucción Si para diseñar estructuras de selección que contengan más de dos alternativas. Por ejemplo, una estructura Si-entonces puede contener otra estructura Si-entonces, y esta estructura Si-entonces puede contener otra, y así sucesivamente cualquier número de veces; a su vez, dentro de cada estructura pueden existir diferentes acciones, a esto se le llama condiciónales anidados o en cascada. Pseudocódigo en PSeInt: Si expresion_logica1 Entonces acciones_por_verdadero1 Sino Si expresion_logica2 Entonces acciones_por_verdadero2 Sino Si expresion_logica4 Entonces acciones_por_verdadero3 Sino acciones_por_falso Fin Si Fin Si Fin Si Ejemplo para abrir en Pseint: Condicionales Anidados

ESTRUCTURAS REPETITIVAS

Durante el proceso de creación de programas, es muy común, encontrarse con que una operación o conjunto de operaciones deben repetirse muchas veces. Para ello es importante conocer las estructuras de algoritmos que permiten repetir una o varias acciones, un número determinado de veces. Las estructuras que repiten una secuencia de instrucciones un número determinado de veces se denominanbucles, y se denomina iteración al hecho de repetir la ejecución de una secuencia de acciones. Todo bucle tiene que llevar asociada una condición, que es la que va a determinar cuándo se repite el bucle y cuando deja de repetirse. Hay distintos tipos de bucles: • Mientras • Hacer Mientras • Para 5

ESTRUCTURA MIENTRAS

Esta estructura repetitiva Mientras, es en la que el cuerpo del bucle se repite siempre que se cumpla una determinada condición. Cuando se ejecuta la instrucción mientras, la primera cosa que sucede es que se evalúa la condición (una expresión lógica). Si se evalúa falsa, no se toma ninguna acción y el programa prosigue con la siguiente instrucción. Si la expresión lógica es verdadera, entonces se ejecuta el cuerpo del bucle, después de lo cual se evalúa de nuevo la expresión lógica. Este proceso se repite una y otra vez mientras la expresión lógica (condición) sea verdadera, para salir del bucle la condición debe ser falsa. Pseudocódigo en PSeInt: Mientras expresion_logica Hacer secuencia_de_acciones Fin Mientras Regla práctica Las pruebas o test en las expresiones lógica es conveniente que sean mayor o menor que en lugar de pruebas de igualdad o desigualdad. En el caso de la codificación en un lenguaje de programación, esta regla debe seguirse rígidamente en el caso de comparación de números reales, ya que como esos valores se almacenan en cantidades aproximadas las comparaciones de igualdad de valores reales normalmente plantean problemas. Siempre que realice comparaciones de números reales use las relaciones <, <=, > o >=. Ejemplo para abrir en Pseint: Mientras

ESTRUCTURA HACER- MIENTRAS

Esta estructura es muy similar a la anterior, sólo que a diferencia del Mientras el contenido del bucle Hacer-Mientras se ejecuta siempre al menos una vez, ya que la evaluación de la condición lógica se

encuentra al final del bucle. De esta forma garantizamos que las acciones dentro de este bucle sean llevadas a cabo al menos una vez, incluso aunque la expresión lógica sea falsa. 6 Pseudocódigo en PSeInt: Hacer secuencia_de_acciones Mientras Que expresion_logica Regla práctica El bucle hacer-mientras se termina de ejecutar cuando el valor de la condición es falso. La elección entre un bucle mientras y un bucle hacer-mientras depende del problema de cómputo a resolver. En la mayoría de los casos, el bucle mientras es la elección correcta. Por ejemplo, si el bucle se utiliza para recorrer una lista de números (o una lista de cualquier tipo de objetos), la lista puede estar vacía, en cuyo caso las sentencias del bucle nunca se ejecutarán. Si se aplica un bucle hacer-mientras nos conduce a un código de errores. Ejemplo para abrir en Pseint: MientrasQue

ESTRUCTURA PARA

La estructura Para es un poco más compleja que las anteriores y nos permite ejecutar un conjunto de acciones, para cada paso de un conjunto de elementos. Su implementación depende del lenguaje de programación, pero en términos generales podemos identificar tres componentes: la inicialización, finalización y el incremento. La estructura Para comienza con un valor inicial de una variable llamada índice y las acciones especificadas se ejecutan x cantidad de veces, hasta que el valor índice llegue al valor final, a menos que el valor inicial sea mayor que el valor final. La variable índice se incrementa en uno y si este nuevo valor no excede al final, se ejecutan de nuevo las acciones. Por consiguiente, las acciones específicas en el bucle se ejecutan para cada valor de la variable índice desde el valor inicial hasta el valor final con el incremento de uno en uno.

7 Pseudocódigo en PSeInt: Para variable_numerica<-valor_inicial Hasta valor_final Con Paso paso Hacer secuencia_de_acciones Fin Para El incremento de la variable índice (variable_numerica) siempre es 1 si no se indica expresamente lo contrario en el valor de con paso. Dependiendo del tipo de lenguaje, es posible que el incremento sea distinto de uno, positivo o negativo. La variable índice o de control (variable_numerica) normalmente será de tipo entero y es normal emplear como nombres las letras i, j, k. Si el valor_inicial de la variable índice es menor que el valor_final, los incrementos, es decir los pasos, deben ser positivos, ya que en caso contrario la secuencia de acciones no se ejecutaría. De igual modo, si el valor_iniciales mayor que el valor_final, el paso debe ser en este caso negativo, es decir, decremento. Ejemplo para abrir en Pseint: Para Funciones Pseint Además de empezar a implementar las estructuras de control, vamos a empezar a utilizar las funciones de Pseint. Las funciones, son herramientas que nos proporciona Pseint y cumplen el propósito de ayudarnos a resolver ciertos problemas. Supongamos que tenemos que calcular la raíz cuadrada de un numero, Pseint cuenta con una función que pasándole un numero, nos devuelve el resultado de su raíz cuadrada. Ese resultado que devuelve, se lo podemos asignar a una variable o lo podemos concatenar con un escribir para mostrar el resultado sin la necesidad de una variable. También, las funciones se pueden utilizar dentro de cualquier expresión, de cualquier estructura, y cuando se evalúe la misma, se reemplazará por el resultado correspondiente. Tenemos dos tipos de funciones, funciones matemáticas y funciones de cadenas de texto. Las funciones matemáticas, reciben un sólo parámetro de tipo numérico y devolverán un solo valor de tipo numérico. Las funciones de cadenas, en cambio, reciben un solo parámetro de tipo cadena, pero pueden devolver un valor de tipo cadena o de tipo numérico según la función que se use.

8 Funciones Matematicas Función Significado RC(número) Devuelve la raíz cuadrada del número. ABS(número) Devuelve el valor absoluto del número LN(número) Devuelve el logaritmo natural del número EXP(número) Devuelve la función exponencial del número. SEN(número) Devuelve el seno de número. COS(número) Devuelve el coseno de número. TAN(número) Devuelve la tangente de

número. ASEN(número) Devuelve el arcoseno de numero. ACOS(número) Arcocoseno de x ATAN(número) Arcotangante de x MOD Devuelve el módulo (resto de la división entera). TRUNC(número) Trunca el valor x (parte entera de x) REDOND(número) Redondea al valor más cercano a x AZAR(número) Entero aleatorio entre 0 y x -1 ALEATORIO(min,max) Entero aleatorio entre valor mínimo y máximo Ejemplos: Escribir "Raíz cuadrada de 9: " rc(9) Escribir “Resto de 4/2: ” 4 MOD 2 Escribir "Valor absoluto de -3: " abs(-3) Escribir "Seno de 90 grados: " sen(90 * PI / 180) Escribir "Truncamos 3.7: " trunc(3.7) Escribir "Redondeamos 2.7: " redon(2.7) Escribir "Un número al azar del 0 al 9: " azar(10) Escribir "Un número al azar entre 10 y 20: " aleatorio(10,20) Del codigo anterior los resultados serían: Raíz cuadrada de 9: 3 Resto e 4/2: 0 Valor absoluto de -3: 3 Seno de 90 grados: 1 Truncamos 3.7: 3 Redondeamos 2.7: 3 Un número al azar del 0 al 9: 6 Un número al azar entre 10 y 20: 14

9 Funciones Cadenas de Texto Algunas funciones de cadenas de texto, utilizan las posiciones de cada letra de una cadena. Esto significa que, si tengo la palabra Hola, la cadena tendrá 4 posiciones, en Pseint las posiciones de las letras arrancan en 0. Entonces para la cadena Hola, nuestras posiciones serían: 0: H, 1: o, 2: l y 3: a. Función Significado Longitud(cadena) Devuelve la cantidad de letras que compone la cadena. Mayusculas(cadena) Devuelve una copia de la cadena con todas sus letras en mayúsculas. Minusculas(cadena) Devuelve una copia de la cadena con todas sus letras en minúsculas. Subcadena(cadena, posición_inicial, posición_final) Devuelve una nueva cadena que consiste en la parte de la cadena que va desde la posición pos_inicial hasta la posición pos_final. Concatenar(cadena, cadena2) Devuelve una nueva cadena que resulta de unir las cadenas cadena1 y cadena2. ConvertirANumero(cadena) Recibe una cadena compuesta de numeros y devuelve la cadena como una variable numerica. ConvertirACadena(cadena) Recibe un numero y devuelve una variable cadena de caracteres de dicho numero. Ejemplos: Definir cadena1,cadena2,cadena3 como cadena cadena1 = "programacion cadena2 = “EGG” Escribir "La longitud de cadena1 es " longitud(cadena1) Escribir "El primer carácter de cadena1 es " subcadena(cadena1,0,0) Escribir "La cadena1 en mayúsculas es " mayusculas(cadena1) Escribir "La cadena2 en minusculas es " minusculas (cadena2) cadena3 = concatenar(cadena1," es muy interesante") Escribir "La cadena concatenada queda como: "Escribir “La cadena convertida a numero queda:” convertirANumero("10") Escribir “El numero convertido a cadena queda:” convertirATexto(20) Del codigo anterior los resultados serían: La longitud de cadena1 es 12 El primer carácter de cadena1 es p La cadena1 en mayúsculas es PROGRAMACION La cadena2 en minúsculas es egg programacion es muy interesante La cadena convertida a numero queda: 10 El numero convertido a cadena queda: 20 Ejemplo para abrir en Pseint: FuncionesCadenas