Técnico en

DESARROLLO DE SOFTWARE

Lógica de Programación I

Lógica de Programación I

Unidad II

1. Algoritmo

Como vimos en la semana anterior, un algoritmo es una secuencia ordenada de instrucciones finitas que deben seguirse para resolver un problema. En esta unidad estaremos practicando dos tipos de algoritmos y para ello veremos la forma en que se representan los algoritmos.

Formas de representar los Algoritmos

Algoritmos en Lenguaje Natural

Este tipo de algoritmo es el que nos permite visualizar un problema con el lenguaje con el que mejor nos comunicamos. Estos tienden a ser detallados y ambiguos; rara vez son utilizados para representar o visualizar problemas complejos o técnicos.

El ejemplo de algoritmo que les mostramos al iniciar el curso está representado en lenguaje natural. A continuación otro ejemplo en lenguaje natural.

Ejemplo: Algoritmo para cambiar una llanta.

1. Si el carro va en movimiento, orillar el carro y colocar las luces de emergencia.
2. Si el carro esta encendido, apagarlo.

3. Si cuenta con un cono o triángulo para señalizar que el automóvil se encuentra detenido en el camino, de lo contrario continúe con el proceso de cambio de llanta.
4. Si cuenta con las herramientas necesarias para el cambio de la llanta, sáquelas; de lo contrario solicite ayuda para conseguir las herramientas o que alguien más cambie la llanta.
5. Si usted va a cambiar la llanta, afloje los birlos/tuercas/"chuchos" de la llanta a cambiar.
6. Levantar el carro con el gato hidráulico/tricket hasta que la llanta pueda ser retirada sin mayor presión y tomando en cuenta que quepa la llanta inflada.
7. Quitar las tuercas de la llanta a cambiar.
8. Quitar la llanta pinchada.
9. Colocar la llanta pinchada debajo del carro, situándola cerca del gato hidráulico, esto para que la llanta resista al carro en caso de que el gato hidráulico se haya colocado en mal punto.
10. Sacar la llanta de repuesto.
11. Colocar la llanta de repuesto.
12. Apretar las tuercas en la llanta de repuesto que ya se colocó.
13. Bajar el carro y quitar el gato hidráulico.
14. Guardar los repuestos y la llanta pinchada.
15. Guardar el triangular o cono que se haya colocado para prevenir accidentes.
16. Si el carro se encontraba en movimiento previo a fijarse que estaba pinchada la llanta; entonces subirse al carro, encenderlo y continuar con el destino.

Cuando se quiere resolver problemas más complejos mediante algoritmos, es un poco más difícil y largo llevarlo a cabo con con los algoritmos en lenguaje natural; para ello utilizamos los algoritmos simbólicos y estructurados los cuales veremos a continuación.

2. Algoritmos Estructurados

Este tipo de algoritmos se llevan a cabo en base a una colección de reglas, de tal forma que sean entendibles y fáciles de modificar.

Algunas de las reglas que se deben de seguir para crear este tipo de algoritmos son las siguientes:

- El algoritmo debe ser **finito**. Debe tener un número finito de pasos.
- Las instrucciones deben ser **entendibles**.
- Debe llevarse a cabo de tal forma que en un futuro se puedan **entender** tanto por la persona que lo hizo como por cualquier otra persona.
- Los algoritmos y programas que se lleven a cabo en base a los algoritmos deben llevarse a cabo en la **menor cantidad de pasos** y en el **menor tiempo posible**.
- El problema debe dividirse en pequeños módulos.

Para desarrollar estos algoritmos, vamos a utilizar las estructuras de control que vimos en la unidad anterior. Recordemos que una estructura de control es una serie de declaraciones, instrucciones y condiciones necesarias para determinar el flujo dentro de un programa. En la primera unidad vimos las estructuras de control condicionales e iterativas, pero en esta unidad también agregaremos la estructura de control secuencial, la cual no mencionamos la semana pasada debido a lo sencilla que es.

Estructuras de control secuenciales

Cada acción se lleva a cabo una vez en un orden específico.

Estructuras de control condicionales

Permiten seleccionar una acción a realizar entre varias alternativas. Se utilizan para la toma de decisiones.

Estructuras de control iterativas

Una determinada acción se lleva a cabo más de una vez. Se utilizan para repetir un bloque de instrucciones N veces, donde N >= 0.

Pseudocódigo

Es una mezcla entre el lenguaje natural y el lenguaje de programación, el cual es utilizado para plantear un problema y generar una solución lógica y estructurada previo a dar una solución formal en un lenguaje de programación.

En la creación de pseudocódigo, utilizaremos algunas instrucciones aprendidas en la unidad anterior, pero con una mezcla de lenguaje natural en español.

3. Variables en pseudocódigo

Para definir una variable llamada “edad” con valor 10, vamos a hacerlo de la siguiente forma:

```
definir edad ← 10
```

Si ya definimos una vez la variable, no tenemos que estar definiendo la variable cada vez que la utilicemos. De hecho podremos definir todas las variables al inicio del algoritmo y luego simplemente utilizarlas. Se recomienda que se les coloque el tipo de dato que se está definiendo, por ejemplo:

```
definir nombre, apellido:caracter
definir edad, resultado:entero
```

Para realizar operaciones numéricas, vamos a hacerlo de la siguiente forma:

```
resultado ← operador1 + operador2
resultado ← operador1 - operador2
resultado ← operador1 x operador2
resultado ← operador1 / operador2
```

Ejemplo:

```
resultado ← 10 + 15
```

4. Estructura secuencial en pseudocódigo

Como les indicamos con anterioridad, en la estructura secuencial, cada acción se lleva a cabo una vez en un orden específico en un flujo de arriba hacia abajo y sin iteraciones.

Ejemplo para la suma de dos números enteros los cuales se solicitan al usuario:

```
INICIO
  definir numero1, numero2, resultado: entero
  numero1 ← leer numero
  numero2 ← leer numero
  resultado ← numero1 + numero2
  imprimir "El resultado de la suma es: " resultado
FIN
```

5. Estructura condicional en pseudocódigo

```
INICIO
  definir variables:tipodato
  SI condición se cumple ENTONCES
    bloque de código a ejecutar si se cumple la condición.
  SINO
    bloque de código a ejecutar sino se cumple la condición.
  FIN SI
FIN
```

Ejemplo para la división de dos números enteros los cuales se solicitan al usuario:

```
INICIO
   definir numero1, numero2, resultado:entero
   numero1 ← leer numero
   numero2 ← leer numero
   //Tomando en cuenta que la operación será de la forma: numero1/numero2
   SI numero2 es menor o igual a 0 ENTONCES
      imprimir “Error, el divisor debe ser mayor a 0”
   SINO
      resultado ← numero1 / numero2
      imprimir “El resultado es: “ resultado
   FIN SI
FIN
```

6. Estructura iterativa en pseudocódigo

For

```
INICIO
   definir variables:tipodato
   PARA (contador, mientras una condición se cumpla, modificamos contador para evitar ciclos infinitos)
      bloque de código a ejecutar mientras se cumpla la condición
   FIN PARA
```

```
FIN
```

While

```
INICIO
   definir varialbes:tipodato
   MIENTRAS (condición a cumplirse para entrar a la iteración)
bloque de código a ejecutarse mientras se cumpla la condición
   FIN MIENTRAS
FIN
```

Ejemplo para la suma de los números de 1 a 10.

```
INICIO
   definir suma, contador:entero
   suma ← 0
   PARA (contador ← 1, es contador menor o igual a 10?, contador ← contador +
1)
      suma ← suma + contador
   FIN PARA
   imprimir “El resultado es: “ suma
FIN
```

El mismo ejemplo anterior, pero en lugar de un “For”, utilizaremos un “While” en pseudocódigo.

```
INICIO
  definir suma, contador:entero
  suma ← 0
  contador ← 1
  MIENTRAS (contador es menor o igual a 10)
    suma ← suma + contador
    contador ← contador + 1
  FIN MIENTRAS
  imprimir “El resultado es: “ suma
FIN
```

Referencias

- Therold E. Bailey, Kris Lundgaard. Program Design With Pseudocode. Brooks/Cole. 1989.

Descargo de responsabilidad

La información contenida en este documento descargable en formato PDF o PPT es un reflejo del material virtual presentado en la versión online del curso. Por lo tanto, su contenido, gráficos, links de consulta, acotaciones y comentarios son responsabilidad exclusiva de su(s) respectivo(s) autor(es) por lo que su contenido no compromete al área de e-Learning del Departamento GES o al programa académico al que pertenece.

El área de e-Learning no asume ninguna responsabilidad por la actualidad, exactitud, obligaciones de derechos de autor, integridad o calidad de los contenidos proporcionados y se aclara que la utilización de este descargable se encuentra limitada de manera expresa para los propósitos educacionales del curso.