Programación.

Valeria Ortiz Cervantes

31 de marzo del 2019

Universidad Nacional Autónoma de México.
Facultad de Ciencias.

Práctica 3. Preguntas.

Datos primitivos.

Concepto:
Los tipos de datos simples o primitivos son los que no están compuestos por otras estructuras de datos, y tienen como característica común que cada variable representa a un elemento.

1. Java:
   (a) Byte: es del tipo entero y representa un dato de 8 bits con signo, ocupa 1 byte, de tal manera que puede almacenar los valores numéricos del -128 al 127.
   (b) Short: es del tipo entero y representa un dato de 16 bits con signo, ocupa 2 bytes, de manera que almacena los valores numéricos del -32 768 al 32 767.
   (c) Integer: es del tipo entero y es un dato de 32 bits con signo, ocupa 4 bytes, por lo cual almacena los valores hasta el 2x109, corresponden al conjunto de los números enteros.
   (d) Long: es del tipo entero y es un dato de 64 bits con signo, ocupa 8 bytes.
   (e) Float: es del tipo decimal simple y es un dato para almacenar números en coma flotante con precisión simple de 32 bits, ocupa 4 bytes, corresponden al conjunto de los números reales.
   (f) Double: es del tipo decimal doble y es un dato para almacenar números en coma flotante con doble precisión de 64 bits, ocupa 8 bytes.
   (g) Boolean: es para definir tipos de datos booleanos, es decir, aquellos que tienen un valor de true o false con 1 bit de información, ocupa 1 byte.
   (h) Char: es un tipo de dato que representa a un carácter unicode sencillo de 16 bits, ocupa 2 bytes.
2. Python:
   (a) Integer
   (b) Long (solo en Python 2, en Python 3 ya no)
   (c) Float
   (d) Boolean
   (e) Complex: corresponden al conjunto de los números complejos, se compone de dos números de tipo Float separados por el operador de adición "+", en el que el primer número corresponde al componente en los números reales y el componente en los números imaginarios es identificado añadiéndole la letra "j" al final.

(f) String: son secuencias de caracteres delimitados por comillas (“ ”) o apostrofes (‘ ‘).

(g) None: este tipo representa un valor “vacío”.

3. C++:

   (a) Char
   (b) Integer
   (c) Short Integer
   (d) Long Integer
   (e) Boolean
   (f) Float
   (g) Double
   (h) Long Double

4. C#:

   (a) Integer
   (b) Float
   (c) Double
   (d) Char
   (e) Boolean

5. R:

   (a) Integer
   (b) Complex
   (c) Logical
   (d) Matrix
   (e) String

6. Javascript:

   (a) Boolean
   (b) Null
   (c) Undefined
   (d) Number
   (e) String
   (f) Symbol

Algoritmos y su complejidad.

1. Problema: Ordenar los elementos de una lista.
   Algoritmo: Quicksort

   (a) Elegir al azar un elemento de la lista (el pivote).

   (b) Reordenar los demás elementos de la lista a cada lado del pivote, de manera que a un lado queden todos los menores que él y al otro lado los mayores.

   (c) La lista se separa en dos sublistas, una formada por los elementos a la izquierda del pivote y otra por los elementos a la derecha.

   (d) Se repite el proceso de forma recursiva para cada sublista mientras éstas contengan más de un elemento.

   Complejidad:

   (a) En el mejor caso, el pivote termina en el centro de la lista, dividiéndola en dos sublistas del mismo tamaño. En este caso el orden de complejidad del algoritmo es O(n log(n)), donde n es el tamaño de la entrada.

   (b) En el peor caso, el pivote termina en un extremo de la lista, entonces el orden de complejidad será de O(n2). Usualmente ocurre en listas que se encuentran ordenadas o casi ordenadas.

   (c) En el caso promedio la complejidad es de O(n log(n)).n el caso promedio la complejidad es de O(n log(n)).

2. Problema: Obtener el siguiente término de la sucesión de Fibonacci.
   Algoritmo:

   (a) Si hacemos una programación iterativa de manera en que la entrada sean dos términos de la sucesión, se realice la suma y te devuelva el siguiente término la complejidad será de O(n), siendo n el número de sumas totales.

   (b) ii. Si consideramos las potencias de la matriz

   $$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

   es fácil demostrar por inducción que están relacionadas doblemente con la sucesión de Fibonacci, por la igualdad

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} = \begin{bmatrix} F_n & F_{n-1} \\ F_{n-1} & F_{n-2} \end{bmatrix}$$

Por tanto, aplicando el algoritmo de la exponenciación rápida a la matriz es posible calcular el n-ésimo término de la sucesión en log_2(n) iteraciones, por lo que la complejidad sería de O(log_2(n))

3. Problema: Calcular el máximo común divisor de dos números.
   Algoritmo: Utilizaremos el algoritmo de Euclides.
   (a) Si hacemos una función recursiva que tome en cuenta el residuo de la división de ambos números.
   (b) En general, el número de divisiones efectuadas por el algoritmo nunca supera 5 veces el número de dígitos que tienen estos números.
   (c) Lo cual significa que la complejidad será de O(log n), siendo n el mayor de los números.