



Estructuras de datos

Arle Morales Ortiz

Docente

Ingeniería de software



PROGRAMA DE
**INGENIERÍA
DE SOFTWARE**
CUE AVH *Dual*

Estructura de datos



De acuerdo con Martín Quetglás, 2002, la mayor parte de la información útil en la práctica “*no aparece en forma de datos simples aislada de otro tipo de datos, al contrario, aparece de forma estructurada y organizada. Las enciclopedias, diccionarios, revistas, libros en general, son colecciones de datos que serían complejos por no decir imposibles de leer si no hicieran parte de una organización lógica con determinadas reglas*”.

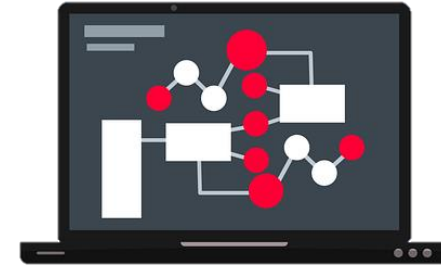
Estructura de datos



Por consiguiente, el agrupar la información y poner en ella una estructura facilita su acceso, administración y hace aún más importante y relevante su contenido. Por ello, afirma (Echeverry, 2014) “la importancia de la escogencia de una estructura de datos frente a otra, ya que en el momento de la programación es decisivo el algoritmo a utilizarse para la resolución de determinado problema recordando siempre la ecuación”:

"PROGRAMACIÓN = ESTRUCTURA DE DATOS + ALGORITMOS"

Estructura de datos



A esto se le suma, expresa (Joyanes, 2006) “Se estudian estructuras de datos con un objetivo fundamental: *aprender a escribir programas más eficientes*. La potencia de cálculo y las capacidades de almacenamiento aumentan la eficacia y ello conlleva un aumento de los resultados de las máquinas y de los programas desarrollados por ellas”. El desarrollo de programas eficientes no tiene que ver con “trucos de programación” sino, que se apoyan en una excelente estructuración de la información y buenos algoritmos.

Recursividad

Es aquella propiedad que posee un método por la cual puede llamarse a sí mismo.

Ejemplo 1: Algoritmo recursivo de la función matemática que suma los n primeros números enteros positivos.

2

El algoritmo que determina la función modo recursivo ha de tener presente de salida o una condición de parada.

```
1 package com.cue.ed.clases;
2
3 public class Sumatoria {
4
5     public long sumaNenteros(int n) {
6         if (n == 1)
7             return 1;
8         else
9             return n + sumaNenteros(n - 1);
10    }
11 }
```

1

Se está definiendo la función o método respecto de sí mismo.

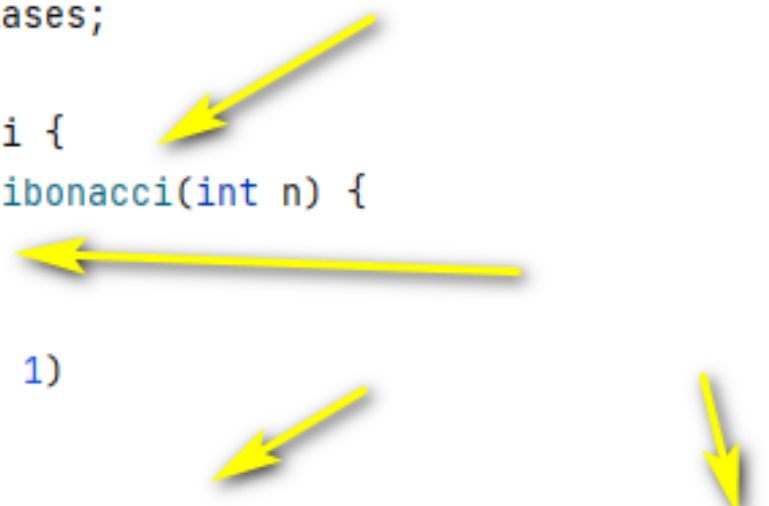
implementación

```
1 package com.cue.ed.principal;
2
3 import com.cue.ed.clases.Sumatoria;
4 import java.util.Scanner;
5
6 public class PrincipalSumatoria {
7     public static void main(String[] args) {
8         Scanner sc = new Scanner(System.in);
9         Sumatoria sumatoria = new Sumatoria();
10
11         System.out.print("Ingrese número de veces: ");
12         System.out.println("Resultado " + sumatoria.sumaNenteros(sc.nextInt()));
13     }
14 }
```

Ejemplo 2: Definir la naturaleza recursiva de la serie de Fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, 21 ...

Se observa en esta serie que comienza con 0 y 1, y tiene la propiedad de que cada elemento es la suma de los dos elementos anteriores.

```
1 package com.cue.ed.clases;
2
3 public class Fibonacci {
4     public int printFibonacci(int n) {
5         if (n == 0)
6             return 0;
7         else if (n == 1)
8             return 1;
9         else
10            return printFibonacci(n - 1) + printFibonacci(n - 2);
11     }
12 }
```




```
1 package com.cue.ed.principal;
2
3 import com.cue.ed.clases.Fibonacci;
4
5 import java.util.Scanner;
6
7 public class PrincipalFibonacci {
8     public static void main(String[] args) {
9         Scanner sc = new Scanner(System.in);
10        Fibonacci fi = new Fibonacci();
11        System.out.println("máximo : ");
12        int num = sc.nextInt();
13        for (int i = 0; i <= num; i++) {
14            int fibonacciNumber = fi.printFibonacci(i);
15            System.out.print(" " + fibonacciNumber);
16        }
17    }
18 }
```


Ejemplo 3: escribir un método recursivo que calcule el factorial de un número n y un programa que pida un número entero y escriba su factorial.

```
1 package com.cue.ed.clases;
2
3 public class Factorial {
4     public long calcularFactorial(int n) {
5         if (n <= 1)
6             return 1;
7         else {
8             long resultado = n * calcularFactorial(n - 1);
9             return resultado;
10        }
11    }
12 }
```

```
1 package com.cue.ed.principal;
2
3 import com.cue.ed.clases.Factorial;
4
5 import java.util.Scanner;
6
7 public class PrincipalFactorial {
8     public static void main(String[] args) {
9         Scanner sc = new Scanner(System.in);
10        Factorial factorial = new Factorial();
11
12        System.out.println("número factorial : ");
13        System.out.println("resultado " + factorial.calcularFactorial(sc.nextInt()));
14    }
15 }
```

Ejemplo 4: Mostrar por consola el alfabeto, utilizando recursión indirecta.

```
1 package classes;
2
3 public class Alfabeto {
4     public static void main(String[] args) {
5         System.out.println();
6         metodoA('Z');
7         System.out.println();
8     }
9
10    static void metodoA(char c) {
11        if (c > 'A')
12            metodoB(c);
13        System.out.print(c + " ");
14    }
15
16    static void metodoB(char c) {
17        metodoA(--c);
18    }
19 }
```

Ejemplo 5: Recursión de cadena inversa Java.

```
1 package clases;
2
3 public class String_Revers {
4     public void reverseString(String cadena) {
5         if ((cadena == null) || (cadena.length() <= 1))
6             System.out.println(cadena);
7         else {
8             System.out.print(cadena.charAt(cadena.length() - 1));
9             reverseString(cadena.substring(0, cadena.length() - 1));
10        }
11    }
12 }
```

```
1 package principal;
2
3 import clases.String_Revers;
4
5 public class PrincipalCadena {
6     public static void main(String[] args) {
7         String inputcadena = "esto es un testing de software";
8         System.out.println("cadena original: " + inputcadena);
9
10        String_Revers obj = new String_Revers();
11        System.out.print("cadena invertida: ");
12        obj.reverseString(inputcadena);
13    }
14 }
```

Ejercicio 1:

1. Implementar un programa en java que use un método recursivo para verificar si una palabra es palíndromo.

Ejemplo: Somos o no somos, Isaac no ronca así

2. Implementar un programa en java que use un método recursivo para encontrar el valor mínimo en un arreglo.

```
int numArray[] = { 7, 32, 64, 2, 10, 23 };
```

3. Implementar un programa en java que use un método recursivo para encontrar los números positivos y negativos de un arreglo .

4. Implementar el código, hacer debug registrarlo en una tabla en Excel y decir que hace.

```
1  import java.util.*;
2  class Binary_Search {
3      int binarySearch(int numArray[], int left, int right, int key)  {
4          if (right >= left) {
5              int mid = left + (right - left) / 2;
6              if (numArray[mid] == key)
7                  return mid;
8              if (numArray[mid] > key)
9                  return binarySearch(numArray, left, mid - 1, key);
10             return binarySearch(numArray, mid + 1, right, key);
11         }
12         return -1;
13     }
14 }
15 class Main{
16     public static void main(String args[])  {
17         Binary_Search ob = new Binary_Search();
18         int numArray[] = { 4,6,12,16,22};
19         System.out.println("The given array : " + Arrays.toString(numArray));
20         int len = numArray.length;           //length of the array
21         int key = 16;                          //key to be searched
22         int result = ob.binarySearch(numArray, 0, len - 1, key);
23         if (result == -1)
24             System.out.println("Element " + key + " not present");
25         else
26             System.out.println("Element " + key + " found at index " + result);
27     }
28 }
```

5. Implementar el código, hacer debug registrarlo en una tabla en Excel y decir que hace.

```
1 function pow(x, n) {  
2     if (n == 1) {  
3         return x;  
4     } else {  
5         return x * pow(x, n - 1);  
6     }  
7 }  
8  
9 alert( pow(2, 3) ); // 8
```

```
1 using System;  
2 public class CalcularPotenciaRecursiva  
3 {  
4     public static void Main(string[] args)  
5     {  
6         int numero = Convert.ToInt32(Console.ReadLine());  
7         int exponente = Convert.ToInt32(Console.ReadLine());  
8  
9         Console.WriteLine(Potencia(numero, exponente));  
10    }  
11  
12    public static int Potencia(int numero, int exponente)  
13    {  
14        if (exponente == 0)  
15        {  
16            return 1;  
17        }  
18        else  
19        {  
20            return numero * Potencia(numero, exponente - 1);  
21        }  
22    }  
23 }
```

¡Gracias!

Nombre del expositor

Año de la presentación

