3. Introducción a la programación estructurada

En la sección pasada vimos que las sentencias, como las imperativas o declarativas, son elementos de todo programa. Vimos, en particular, la sintaxis y la semántica de las sentencias de asignación; comenzaremos esta sección, hablando sobre las sentecias de control.

Una **sentencia de control** altera la secuencia de ejecución de un programa. Los lenguajes de programación de hoy en día están diseñados con sentencias de control que permitan a todo un patrón de ramificación ser expresado a través de una sóla estructura léxica. El objetivo es que el lenguaje no sólo permita que los algortimos sean expresados en una forma más legible, sino que también asistan al programador en implementar dicha legibilidad. El resultado es la práctica conocida como programación estructurada cuyos elementos son

- las estructuras de control
- las funciones y
- los bloques.

Pero iremos paso a paso, nos centraremos sólo en 3 de los 4 cuatro tipos de estructuras de control. Para ello utilizaremos scratch, un lenguaje de programación visual desarrollado por el MIT Media Lab. Finalmente, por una cuestión de completitud, describiremos superficialmente el resto de elementos de la programación estructurada.

3.1. Estructuras de control

Encontraremos cuatro tipos de estructuras de control:

- Estructura secuencial
- Estructura selectiva
- Estructura iterativa
- Estructura recursiva

3.2. Funciones

Una **función** en el contexto de la programción estructurada es una secuencia de sentencias que pueden ser ejecutadas por una sóla sentencia al ser referida o evocada. Por ejemplo, en el siguiente programa, escrito en el lenguaje de programación C:

- en la línea 9 del código se evoca a la función sumar y

- de la línea 13 a la 16 se define a la función sumar.

```
1  // prototipo de la funcion sumar
2  int sumar(int x, int y);
3
4  int main ()
5  {
6          int x, y; // sentencia declarativa
7          x = 2; // sentencia de asignacion
8          y = 3;
9          sumar(x,y); // evocando a sumar
10 }
11
12 // definicion de la funcion sumar
13 int sumar(int x, int y)
14 {
15         return x + y;
16 }
```

3.3. Bloques

Un **bloque** es una secuencia de sentencias que es tratada como una sóla sentencia. Por ejemplo, en el siguiente programa, las sentecias de las líneas 8, 9 y 10 se comportan como una sóla al ejecutarse una vez que la expresión dentro de los paréntesis (línea 6) se evalúa como verdadera.

```
1  int main ()
2  {
3          int x, y, aux; // sentencia declarativa
4          x = 2; // sentencia de asignacion
5          y = 3;
6          if (x < y)
7          { // incicio del bloque
8                  aux = x;
9                  x = y;
10                 y = aux;
11         } // fin del bloque
12         x = 5;
13 }
```

Tarea: Leer páginas 139-151 de [Brookshear and Brylow, 2015].

Referencias

[Brookshear and Brylow, 2015] Brookshear, G. and Brylow, D. (2015). *Computer Science - An Overview*. Pearson Education Limited, 12th edition.