2. Conceptos básicos de la programación

El gran desarrollo de sistemas de software cada vez má complejos hubiese sido prácticamente imposible si los humanos hubiesen sido forzados a escribir programas en el lenguaje de las máquinas. Consecuentemente, muchos lenguajes de programación han sido desarrollados de tal manera que permitan a los algoritmos ser expresados en una forma que sea a la vez amigable con los humanos y fácilmente convertibles en instrucciones del lenguaje de las máquinas.

A continuación abordaremos algunos conceptos propios de la programación imperativa. Por lo general, un programa consiste en un conjunto de **sentencias** o intrucciones que caen en una de las tres categorías:

- **sentencias declarativas** que definen la terminología usada en el programa;
- **sentencias imperativas** que describen los pasos en el subyacente algoritmo;
- **comentarios** que mejoran la legibilidad del programa explicando un características ocultas en una forma que podamos entender.

Un programa imperativo puede ser visualizada de la siguiente forma: comienza con un conjunto de sentencias declarativas describiendo los datos a ser manipulados por el programa; luego, es seguido por sentencias imperativas que describen el algoritmo a ser ejecutado; y finalmente, los comentarios dispersos en todo el programa como sean necesarios para clarificarlo. Sigiendo este hilo conductor, comenzamos nuestro estudio de conceptos de programación.

2.1. Variables y tipos de dato

Los lenguajes de alto nivel permiten a los lugares de la memoria de una computadora ser referenciados por nombres descriptivos. Tales nombres son conocidos como **variables**, por la virtud de que al ser cambiado el valor guardado en tal lugar, el valor asociado al nombre cambia conforme el programa se ejecuta. Estas variables deben ser identificadas via un sentencia declarativa que, a su vez, el programador describa el **tipo de dato** a ser guardado en el lugar de la memoria asociado con la variable. Trabajaremos con los siguientes tipos de datos:

```
entero
real
caracter
```

Por ejemplo, si queremos declarar las variables edad, peso y nota como tipos de dato entero, real y caracter, respectivamente, escribiríamos:

```
entero edad
real peso
caracter nota
```

O si queremos declarar las variables ingreso, egreso y utilidad como tipo de dato real, sentenciaríamos:

```
real ingreso, egreso, utilidad
```

2.2. Estructura de datos

Además del tipo de dato, las variables en un programa son a veces asociadas con **estructura de datos** que es la combinación de datos. Por ejemplo:

- un texto pueden ser visualizado como una larga cadena de caracteres,
- un reporte nacional de temperaturas máximas diarias puede ser representado como una tabla rectangular de datos numéricos donde las filas representan el día y las columnas cada ciudad capital de Perú.
- una guía telefónica puede ser identificada como una lista de ítems donde cada ítem tiene el nombre y el número telefónico de cada persona.

Una estructura de datos común es el **arreglo** que es un bloque de elementos del mismo tipo como una

- lista unidimensional,
- una tabla bidimensional con filas y columnas,
- una tabla con más de dos dimensiones.

Por ejemplo, si quiere declarar el arreglo unidimensional nombre de longitud 25, sentenciaría:

```
caracter nombre[25]
```

Y si quiere declarar el arreglo bidimensional temperaturas de 30 filas y 24 columnas, escribiría:

```
real temperaturas[30][24]
```

2.3. Constantes y literales

Algunas veces, valores prederteminados son usados en sendas ocaciones en un programa. Así, si queremos calcular el perímetro y área de 10 circunferencias, podríamos "registrar" el valor de π. Para resolver estos problemas, "asignamos"nombres descriptivos a valores específicos e inmutables. Tal nombre es llamado de **constante**. Por ejemplo, si queremos asignar a la constante PI el valor de 3.14, escribiríamos:

```
1 const real PI = 3.14
```

Por otro lado, la apariencia explícita de un valor es llamado de literal. Así, en el ejemplo de arriba 3.14 es un literal.

2.4. Sentencias de asignación

Una vez que la terminología a ser usada en el programa (tales como variables o constantes) ha sido declarada, procederemos a describir los algoritmos envueltos. La sentencia imperativa más básica de todas es la sentencia de asignación que solicita que un valor sea asignado a una variable (o para ser más precisos, almacenado en la porci´n de la memoria identificada por la varaible).

- La sintaxis de estas sentencias son la de una variable, seguida por un símbolo que representa el operador de asignación (nosotros emplearemos el símbolo =), y luego por una expresión indicando el valor a ser asignado.
- La semántica de estas sentencias es que la expresión es evaluada y el resultado almacenado como el valor de la variable.

Por ejemplo, si queremos asignar a la variable utilidad el valor que resulta de evaluar el valor de la variable ingreso menos el valor de la variable egreso, sentenciaríamos:

```
1 utilidad = ingreso - egreso
```

2.5. Comentarios

Sin importar qué tan bien un lenguaje de programación está diseñado y qué tan bien las características del lenguaje son aplicadas a un programa, información adicional suele ser útil u obligatoria cuando un ser humano trata de leer y entender el programa. Por esta razón, los lenguajes de programación proporcionan maneras de insertar declaraciones explicativas, llamados **comentarios**, dentro de un programa.

Tarea: Leer páginas 276-280 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.