Clase 6: Lógica de programación

Mauricio Gómez García

Índice

Introducción

La lógica de programación es la herramienta que nos permite construir y crear software y código que cumpla las funciones que son requeridas por el programador. Forma la base más fundamental de cualquier lenguaje de programación, y tiene la característica que se puede traspasar entre cualquier lenguaje. Es decir, si tienes bases fuertes de lógica de programación será posible leer, entender y escribir código para cualquier lenguaje.

Esta sesión se dedicará a revisar algunos conceptos básicos de la lógica de programación con ejemplos prácticos utilizando C.

Seccionar instrucciones

Una de las partes más fundamentales para implementar la lógica de programación es tener la habilidad de tomar un problema y seccionarlo o dividirlo en sus partes más fundamentales. Un problema complejo puede tener una solución compleja, pero si se divide en muchos pasos pequeños, cada uno de dichos pasos se puede resolver con soluciones simples que se unen para completar el problema complejo.

Propongamos el siguiente ejemplo: un programa que cumpla una función matemática. Si tenemos la necesidad de procesar una función matemática compleja, dividir cada paso de la función en partes más sencillas nos permitirá crear un programa de manera más fácil. Tomemos la siguiente función:

$$F_x = \left(\frac{32x}{4}\right)^3$$

Observando, existen varios pasos que debemos seguir para obtener un valor con esta función en un programa. Empezemos a separar cada parte:

1) F_x es una función, por lo que debemos crear una nueva función programática para realizar sus instrucciones.
2) x se debe multiplicar por 32.
3) El resultado de la suma se debe dividir sobre 4.

4) El resultado de la división se debe elevar a la tercera potencia (o, más explícitamente, se debe multiplicar por si mismo tres veces).

Observemos el último paso. Este paso se puede dividir aún más para describir los pasos a realizar aún más, creando una nueva función para elevar números a cierta potencia:

1) Tomar el valor a elevar.
2) Tomar el valor de la potencia.
3) Multiplicar el valor de entrada por si mismo la cantidad de veces establecido.
4) Devolver el valor de las multiplicaciones.

Tomando todo esto en cuenta, construyamos un programa en C que realize estos pasos:

```c
#include <stdio.h>

// Declaracion de funciones
int f(int x);
int potencia(int x, int y);

// Funcion principal
int main() {
  int x;
  printf("Introduzca un numero: ");
  scanf("%d", &x);
  printf("El resultado usando %d es %d\n", x, f(x));
  return 0;
}

// Funcion de potencia
int potencia(int x, int y) {
    int resultado = 1;
    for (int i=0; i<y; i++) {
        resultado*=x;
    }
    return resultado;
}

// Funcion matematica
int f(int x) {
    int resultado = x;
    resultado = (32*x)/4;
    resultado = potencia(resultado, 3);
    return resultado;
}
```

La salida de este programa al introducir 5 debe ser:

```
El resultado usando 5 es 64000
```

Pseudo-código

Una herramienta útil mientras se aprende a programar (e incluso ya teniendo experiencia) el pseudo-código, que como su nombre indica es texto o ciertas estructuras que no están escritas en ningún lenguaje de programación existente, pero describen el orden de pasos para cumplir un objetivo.

Propongamos el siguiente ejemplo: un programa que tome un valor y nos devuelva su cuadrado. Para escribir nuestro pseudo-código es necesario dividir el objetivo en pasos para llegar a el. En este ejemplo existen los siguientes pasos:

1) Crear una variable para guardar el valor de entrada.
2) Solicitar el valor de entrada al usuario.
3) Leer el valor de entrada y guardar en nuestra variable.
4) Multiplicar el valor de entrada por si mismo.
5) Guardar el resultado en una nueva variable.
6) Imprimir el valor a la consola.

Estos serían los pasos necesarios a realizar para completar el programa que se busca. Una vez que se entiendan los pasos o instrucciones que se requieren para llegar al objetivo, se puede empezar a escribir pseudo-código que cumple la función:

```
int x
imprimir("Introduzca un numero: ")
leer(x)
x=x*x
imprimir(x)
```

Como se puede observar, nuestro pseudo-código no tiene el formato de ningún lenguaje existente, pero nos puede ayudar para segmentar el orden de las instrucciones a realizar mejor ántes de empezar a escribir nuestro código final. Observemos un ejemplo de código en C que realiza estos mismos pasos:

```c
#include <stdio.h>

void main() {
  int x;
  printf("Introduzca un numero: ");
  scanf("%d", &x);
  int y = x*x;
  printf("El resultado es %d\n",y);
}
```

Este código es muy similar a nuestro psuedo-código, y se tiene la ventaja de poder utilizar el mismo método para escribir código para otros lenguajes. Observemos el siguiente ejemplo escrito en Python:

```python
x = 0
x = int(input("Introduzca un numero: "))
y = x*x
print("El resultado es", y)
```

Al compilar y ejecutar el código de ejemplo en C, observamos que al ingresar un valor numérico nos imprime siempre el valor al cuadrado. Introduciendo el valor 5, por ejemplo, nos devuelve la siguiente salida a la consola:

```
El resultado es 25
```

Conclusión

El código funciona a travez de instrucciones lógicas, y poder implementar la lógica de programacionón asegura poder crear soluciones de software que cumplan el objetivo del desarrollador. Esta habilidad se puede trabajar y fortalecer, por lo que completar ejercicios de práctica para resolver problemas es crucial para entender mejor la lógica de programación.

Existen muchas maneras de segmentar los problemas para convertirlos en pasos lógicos, lo cual permite a los desarrolladores crear programas de software a partir de ellos, y utilizar el más adecuado para cada persona permite que se desarrollen de su propia manera.

Tarea opcional
Se recomienda practicar la lógica de programación completando retos o creando programas para solucionar tareas de la vida cotidiana. Siguiente se enlistan ideas para desarrollar programas que cumplan un objetivo o resuelvan algún problema:

- Una calculadora utilizando las herramientas vistas préviamente, para lograr la suma, resta, multiplicación y división de valores que otorga el usuario.
- Un programa que imprima el texto del usuario n cantidad de veces, introducido por el usuario.
- Un programa que utilize la recursión para calcular factoriales.