Diferencias entre el lenguaje Javascript y el lenguaje C.

Programación I – Laboratorio I.
Tecnicatura Superior en Programación.
UTN-FRA

Autor: *Ing. Ernesto Gigliotti*

Revisores: *Lic. Mauricio Dávila*

Versión : 1

Índice de contenido

1 Diferencias entre el lenguaje Javascript y el lenguaje C

1.1 Sintaxis del lenguaje C

La estructura general de la sintaxis del lenguaje C es prácticamente la misma que la del lenguaje Javascript, se utilizan las mismas palabraas reservadas para los bucles (*while, do-while, for*) se utilizan las mismas palabras reservadas para las sentencias condicionales (*if, if-else, switch, break*) asi como también los mismos operadores de comparación (*==, !=, <, >, <=, >=*).

La utilización de llaves (“{”, “}”) para crear bloques de código y el carácter terminador de línea “;” son reglas que se mantienen en ambos lenguajes.

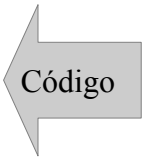
1.2 Punto de entrada de la aplicación

Un programa escrito en lenguaje C, posee una función llamada “main”. Esta función se llamará por el sistema operativo cuando el programa comience, es decir que el código que queremos que se ejecute deberemos colocarlo dentro de esta función:

```
int main(int argsc, char ** argsv) {
    int valorA;
    int valorB;
    int resultado;

    valorA=5;
    valorB=7;
    resultado=valorA+valorB;

}
```



1.3 Variables: Tipos de datos

Una de las diferencias principales es la aparición del concepto de tipo de dato. A cada variable definida se le deberá indicar qué tipo de dato almacenará (número entero, número con decimales, caracteres)

Debido a esto, la definición de una variable donde se almacenarán números se transforma de la sintaxis de Javascript:

```
var valorA;
```

a la sintaxis del lenguaje C:

```
int valorA;
```

Si queremos hacer un programa que sume dos números:

Javascript:

```
var valorA;
var valorB;
var resultado;

valorA=5;
valorB=7;
resultado=valorA+valorB;
```

Lenguaje C:

```
int valorA;
int valorB;
int resultado;

valorA=5;
valorB=7;
resultado=valorA+valorB;
```

Una diferencia que puede observarse es que en el primer ejemplo (Lenguaje Javascript) se podrían sumar números con decimales simplemente cargando en las variables valorA y valorB números con decimales:

```
valorA=5.5;
valorB=7.3;
```

En el caso del lenguaje C, debido a que las variables están definidas como del tipo de dato "int" (número entero) al cargar el valor 5.5, la variable solo almacenará la parte entera, es decir el número 5.

Debido a esta diferencia, el siguiente programa en lenguaje C:

```
int valorA;
int valorB;
int resultado;

valorA=5.5;
valorB=7.3;
resultado=valorA+valorB;
```

daría como resultado en la variable "resultado" el valor 12 (5+7) y no el valor 12.8. Para poder sumar los números con décimas, deberemos cambiar la definición de la variable:

```
float valorA;
float valorB;
float resultado;

valorA=5.5;
valorB=7.3;
resultado=valorA+valorB;
```

En este ejemplo al cambiar el tipo de dato de la variable de "int" a "float" indicamos que la variable tiene la capacidad de almacenar un número con decimales.

La siguiente tabla indica algunos de los tipos de datos del lenguaje:

Palabra reservada	**descripción**
int	Número entero (4 bytes)
short int	Número entero (2 bytes)
long long	Número entero (8 bytes)
float	Número con decimales (4 bytes)
double	Número con decimales (8 bytes)
char	Un carácter (1 byte)
void	Sin tipo

1.4 Operadores aritméticos básicos

Los operadores aritméticos son los mismos que para el languaje Javascript:

- **+** : Suma
- **-** : Resta
- ***** : Multiplicación
- **/** : División
- **%** : Resto

1.5 Salida de datos: Consola

A diferencia de Javascript, un programa en languaje C no se ejecuta dentro de un navegador web, sino que cuando el programa es compilado se genera un ejecutable (*.exe* en Windows, archivo ejecutable en Unix), por lo que la única salida por pantalla que dispondremos (excepto que se utilice una biblioteca gráfica) será la salida por consola del sistema operativo (*cmd* en Windows o *Terminal* en Unix)

En Javascript podíamos imprimir por la consola del navegador utilizando:

```
console.log("Introduccion a Javascript");
```

En lenguaje C utilizaremos la función *printf*:

```
printf("Introduccion a lenguaje C");
```

Cuando queremos imprimir el valor de una variable, ya no podremos utilizar el operador "+" para concatenar texto, sino que deberemos utilizar los **modificadores de formato**:

Javascript:
```
console.log("El resultado es:"+resultado);
```

Lenguaje C:
```
printf("El resultado es: %d",resultado);
```

Los modificadores de formato deben pensarse como un símbolo que será reemplazado por un valor en el momento en que se imprime el mensaje. En el ejemplo anterior, el símbolo "%d" se reemplazará por el valor de la variable "resultado". Si queremos imprimir el valor de más de una variable, estas variables se pasan como argumento a la función printf en el mismo orden que serán reemplazadas por los modificadores de formato:

```
printf("Varible1: %d – Variable2: %d",var1,var2);
```

Existen diferentes modificadores de formatos para indicar de qué forma queremos imprimir el valor de la variable, a continuación se detallan los más utilizados:

Modificador	Descripción
`%d`	Imprime un número entero
`%f`	Imprime un número con decimales
`%c`	Imprime un caracter
`%s`	Imprime un array de caracteres

1.6 Entrada de datos: Consola

Para permitir al usuario ingresar datos, ya no disponemos del prompt de Javascript, deberemos utilizar la función scanf, la cual recibirá un modificador de formato, y la variable donde se almacenará lo ingresado:

```
int valorA;
scanf("%d",&valorA);
printf("Ingresaste: %d",valorA);
```

Cabe remarcar el símbolo "&" antes de la variable, que se tratará en detalle cuando se aborde el tema de punteros.

En el ejemplo anterior, el programa se quedará esperando que el usuario ingresa un número por la consola, al presionar ENTER, se guardará el número ingresado en la variable "valorA".

Para ingresar un texto (*String*) deberemos utilizar el concepto de *arrays*, ya que el lenguaje no posee el tipo de dato *String* como Javascript, por lo que un *String* será un *array* de caracteres (un *array* de variables del tipo *char*)

Definimos un String de la siguiente manera:

```
char texto[128];
```

Deberemos indicar el tamaño máximo que tendrá la variable (en este caso 128 letras).

Luego podremos pedir al usuario que ingrese un texto usando *scanf* y el modificador de formato "%s":

```
scanf("%s",texto);
```

NOTA: Es importante destacar que en el caso de los *arrays*, no se utiliza el símbolo "&".

Cuando el usuario ingrese un texto y presione ENTER, el texto se almacenará en la variable "texto", y podremos imprimirla también utilizando el modificador de formato "%s":

```
printf("Ingresaste: %s",texto);
```

1.7 Operaciones lógicas y control de flujo

En el lenguaje Javascript, teníamos un tipo de dato llamado boolean y las palabras reservadas "true" y "false" que eran los valores que este tipo de variables podían adoptar. En el lenguaje C, no existe dicho tipo de dato, y se utiliza para reemplazarlo el tipo "int".

Esto quiere decir que una comparación lógica que en Javascript nos daba como resultado "true" o "false" ahora dará como resultado "0" o "1":

Operación lógica en Javascript:

```
var a;
var b;
var resultado;

a = true ;
b = false ;

resultado = a && b ;

console.log("Resultado de a intersección b:"+resultado);
```

Equivalente en lenguaje C:

```
int a;
int b;
int resultado;

a = 1 ;
b = 0 ;

resultado = a && b ;

printf("Resultado de a intersección b: %d",resultado);
```

El resultado del código de C no será "false" sino "0". De la misma manera, cualquier comparación entre variables no dará como resultado "true" o "false" sino "1" o "0", por ejemplo:

```
float variableA;
float variableB;
int resultado;
variableA = 3.14;
variableB = 5.5;
resultado = variableA < variableB;
printf("Resultado de la comparación: %d",resultado);
```

En en ejemplo anterior, la variable "resultado" contendrá el valor "1".

1.8 Sentencias de control de flujo

Debido a que no existe el tipo de dato boolean, las sentencias de control de flujo ya no requieren recibir un boolean sino un número. Debido a que generalmente hacemos al conparación dentro de los paréntesis del "if", no notaremos ninguna diferencia con respecto a Javascript.

Javascript:

```
var nota;
nota = 8;

if(nota >= 7)
{
      console.log("El alumno está aprobado");
}
```

Lenguaje C:

```
int nota;
nota = 8;

if(nota >= 7)
{
      printf("El alumno está aprobado");
}
```

Sin embargo es importante destacar que en el primer ejemplo (Javascript) el "if" recibe un dato boolean (resultado de la comparación) mientras que en el segundo ejemplo (Lenguaje C) el "if" recibe un valor numérico (0 o 1) como resultado de la comparación.

1.9 Funciones

El uso de funciones en lenguaje C es igual que en lenguaje Javascript, lo único que cambia es la sintaxis de la definición de la función, pero no la manera de usarla.

Ejemplo de función en Javascript:

```
function calculoCuadrado ( x )
{
      var calculo;
      calculo = x*x;

      return calculo;
}
```

Equivalente en lenguaje C:

```
float calculoCuadrado ( float x )
{
      float calculo;
      calculo = x*x;

      return calculo;
}
```

Como se observa en el ejemplo, la única diferencia es que se deben aclarar los tipos de datos de los argumentos que recibe la función y el tipo de dato del valor que devuelve la función.

En el caso de que la función no devuelva nada o no reciba argumentos, se utilizará la palabra reservada "void"

```
void imprimoMensaje(void)
{
	printf("Hola mundo");
}
```

También se destaca que ya no se utiliza la palabra reservada "function" para indicar que es una función.

1.10 Ubicación de las funciones y prototipos

Para poder usar nuestra función "imprimoMensaje" dentro de la función "main", esta función debe estar escrita previamente:

```
void imprimoMensaje(void)
{
	printf("Hola mundo");
}

int main(int argsc, char ** argsv) {

	imprimoMensaje();
}
```

Si queremos evitar que la función este arriba, podemos declarar un **prototipo** de la misma,y luego colocar la función dodne queramos:

```
void imprimoMensaje(void); // prototipo

int main(int argsc, char ** argsv) {

	imprimoMensaje();
}

void imprimoMensaje(void)
{
	printf("Hola mundo");
}
```

El prototipo solo lleva la primer línea de la función (firma de la función) y un ";" al final.