LECTURA 3: INTRODUCCIÓN A LA PROGRAMACION II

El Pseudocódigo. El pseudocódigo es una forma de escribir los pasos, pero de la forma más cercana al lenguaje de programación que vamos a utilizar, es como un falso lenguaje, pero en nuestro idioma, en el lenguaje humano.

Una de las mayores dificultades con las que se encuentran los hispanoparlantes que empiezan a programar es el idioma. Por eso es bueno utilizar el pseudocódigo, algo así como un falso lenguaje de programación en español, que ayuda a asimilar con más facilidad las ideas básicas.

Por ejemplo si queremos escribir algo en pantalla, en pseudocódigo podríamos poner:
Escribir "Hola" o Escribir 20+30.
También podemos usar:
Mostrar por pantalla "Hola"

Realmente el pseudocódigo lo podríamos escribir como nosotros quisiéramos, ya que realmente no es el programa en sí, solo es una ayuda para posteriormente realizar el programa mediante el lenguaje de programación que utilicemos, eso sí, es de gran ayuda, tanto que es imprescindible. Pero aunque lo podamos escribir de cualquier forma, la mayoría de los programadores suelen usar un vocabulario en común. Este vocabulario será el que veamos aquí.

Recuerda que el pseudocódigo para un programador es fundamental.

Si sabes hacer el pseudocódigo del programa, pasarlo a cualquier lenguaje de programación es muy sencillo, solo tendrás que aprender los comandos equivalentes a las instrucciones en pseudocódigo.

Además, la mayoría de los lenguajes utilizan prácticamente los mismos comandos en su lenguaje.

Sigamos hablando un poco más sobre el pseudocódigo.

Para especificar el principio y el fin del programa pondremos:

Inicio
Aquí iría el programa en pseudocódigo
Fin

Otra forma muy utilizada sería:
Proceso NombreDelPrograma
Aquí iría el programa en pseudocódigo
FinProceso

Las 3 órdenes que más utilizaremos en pseudocódigo son:

Escribir --> Escribe en pantalla el texto que pongamos entre paréntesis o también puede escribir en pantalla el valor de una variable. Esta instrucción en casi todos los programas suele escribirse con la palabra write o document.write('Hola ');.

Leer Edad- -> nos lee desde lo que el usuario marque desde el teclado y guarda el valor, por ejemplo dentro de una variable, en este caso la variable Edad (luego veremos lo que son las variables).

En programación real suele utilizarse la instrucción input.

Calcular 3 x 5 -->Calcula valores

Teniendo el pseudocódigo o el diagrama de flujo lo tenemos muy fácil, ya que es fácilmente traducible a cualquier lenguaje de programación.

Según avancemos en el tema irás viendo ejemplos de pseudocódigo.

Ahora vamos a empezar con lo interesante, vamos a empezar aprender a programar.

Comentarios

Poner comentarios de lo que vamos haciendo es muy útil, sobre todo cuando llega la hora de revisar el programa, si no, más de una vez nos encontraremos diciendo ¿qué hacía esto aquí? No cuesta nada documentar el programa y nos ahorrará dolores de cabeza. La norma que se sigue en todos los programas es poner // delante de los comentarios, para identificarlos:

// Esto será un comentario y no hará nada en el programa
Nosotros durante las explicaciones también pondremos comentarios.

Las variables

Una variable es como una **caja donde metemos** cosas (**datos**). Estos datos **los podemos ir cambiando**, ahora meto un 3, ahora lo quito y meto un 5.

Una variable **tiene un nombre**, que puede ser una letra, una palabra, varias palabras unidas por el guión bajo o varias palabras sin separar pero la primera letra de cada palabra en mayúsculas ejemplo.: VidasPerdidas, vidaperdidas, vidas_perdidas. **Ojo** las mayúsculas y minúsculas son muy importantes en las variables, no es la misma variable **numero** que **Numero**, son dos diferentes. **OJO** tampoco se pueden poner acentos en el nombre de las variables.

Las variables también **tienen un valor** que es **lo que hay dentro de ella** (en la caja) **en ese momento** y que **puede ir variando** según se vaya desarrollando el programa, por eso se llama variable.

Una variable dependiendo de **su valor puede ser numérica**, si solo puede tener un valor numérico, **de texto**, si solo puede contener texto (letra, palabra o frase (string).

En las **variables de texto**, su valor (el texto), debe ir **entre comillas**, para diferenciar que el texto es texto y no es el nombre de otra variable. Por ejemplos vidas = "Cinco" o vidas = "5". En los dos casos el valor es un texto, nunca el valor de 5.

Las numéricas no llevan comillas en su valor. Por ejemplo: vidas = 5. En este caso su valor si que es el número 5.

Hay otras **variables** que se llaman **booleanas** que solo pueden tener dos valores true o false. Normalmente true se puede sustituir por el valor 1 y false por el 0.

Veamos algunos **ejemplos de los tipos de variables**:

Edad=3; **//variable numérica. Fíjate que esto en negrita es un comentario.**
VariableDeTexto= “Tengo 14 años”; //fíjate que va entre comillas.
VariableNumerica= Edad + 2 ; //su valor es el valor de la variable Edad (numérica) +2; en este caso sería = 5 (3+2).
VariableBooleana = true; en este caso sería de valor 1

¿Te has dado cuenta que hemos puesto un punto y coma (;) al acabar de definir cada variable?. En programación **siempre que se acaba una instrucción o grupo de instrucciones se debe poner ";"** para decir al programa que pasamos a otra instrucción diferente. Pero sigamos con las variables.

En algunos lenguajes de programación, lo normal es **declarar las variables** al principio de un programa. Declarar no es más que decir "mira, yo quiero tres variables, y quiero que una se llame Nombre, otra Edad y otra Apellido".

A partir de este momento, podrás meter su valor en cualquier parte del programa.

Por ejemplo, "Juan" en Nombre, un 5 en Edad y "Rodríguez" en Apellido.
Después podrás cambiar su valor, dentro del programa, las veces que quieras.
Ojo no es la misma variable Nombre que nombre, como ya dijimos anteriormente.

Las variables se suelen declarar al principio del programa. OJO RECUERDA que hay lenguajes en los que no es necesario declarar la variable.

Veamos un ejemplo:
numerica: Pepe //declaramos la variable Pepe como numérica.
En Pepe = 14 //Damos un valor a la variable ya declarada Pepe;
Pepe = 42 // Cambiamos el valor de la variable a 42;

Podemos sumar, restar, multiplicar, dividir y hacer cualquier tipo de operación matemática con las variables.

numerica: Pepe, Mari ,Juan //Declaramos las variables que usaremos;
Pepe=2;
Mari=3;
Juan = Pepe + Mari; // Juan tiene ahora el valor numérico de 5.

Los operadores matemáticos más usados en todos los lenguajes de programación (se usan los mismos) son los siguientes:

Operador relacional	*Significado*	*Ejemplo*
>	Mayor que	3>2
<	Menor que	"ABC"<"abc"
=	Igual que	4=3
<=	Menor o igual que	"a"<="b"
>=	Mayor o igual que	4>=5

Hay variables ya definidas por el propio lenguaje de programación que usemos, y **cuyo nombre no se lo podremos dar a ninguna de las que nosotros definamos**.

Las podemos usar pero tal y como el lenguaje las definió. Por ejemplo en muchos lenguajes mouse_x es la variable que tiene el valor de la posición x del ratón en cada momento, hspeed es la velocidad horizontal, etc.

Variables Locales y/o Globales

En muchos lenguajes, si queremos que la variable se use en todo el programa deberemos nombrarla como una **variable global**, en caso contrario, si no la definimos como global, por defecto el lenguaje la considerará una **variable local.**

Una variable local al salir del lugar donde la hemos asignado un valor, perderá ese valor y ya no existirá (al salir de un algoritmo, de un trozo de programa, del objeto, de una estructura IF, etc.).

En la mayoría de los lenguajes se pone la palabra global, un punto y detrás el nombre de la variable, de esta forma, esta variable la podemos usar en todas las partes del programa.

Ejemplo global.pepe, que será distinta de la variable pepe.
Nosotros en todos los ejemplos las usaremos como locales, por eso no verás nunca la palabra global.

Estructuras de control

Las estructuras de control tienen una finalidad bastante definida: su objetivo es ir señalando el **orden en que tienen que sucederse los pasos** de un algoritmo o de un programa.

Las estructuras de control son de tres tipos:

Secuenciales
Selectivas
Repetitivas

Empecemos por las primeras.

Estructuras secuenciales

Una estructura de control secuencial, en realidad, no es más que escribir un paso del algoritmo detrás de otro, el que primero que se haya escrito será el que primero se ejecute.

Veamos un ejemplo: queremos leer el radio de un círculo, calcular su área y mostrar por pantalla al usuario el resultado.

En pseudocódigo sería:

```
numerica: radio, area; //Declaración de variables;

inicio
 Escribir “dame el radio del circulo”;
 Leer radio // asignación del valor de la variable radio por el usuario por medio del teclado;
area =3.14159*radio; //nosotros asignamos el valor de la variable área con su fórmula;
Escribir “el área del circulo es:”    //OJO En los texto SI PODEMOS Y DEBEMOS PONER ACENTOS;
Escribir area; // nos muestra en la pantalla el valor de la variable area resultado de la fórmula anterior;
fin
```

Como ves las instrucciones se van ejecutando unas detrás de otra hasta llegar al final.

Estructuras selectivas

Estas estructuras se utilizan para **TOMAR DECISIONES** (por eso también se llaman estructuras de decisión o alternativas). Lo que se hace es EVALUAR una condición, y, a continuación, en función del resultado, se lleva a cabo una opción u otra.

Alternativas simples (condicional IF)

Son los conocidos "si... entonces". Se usan de la siguiente manera: yo quiero evaluar una condición, y si se cumple (es decir, si es cierta), entonces realizaré una serie de pasos. Un ejemplo

En pseudocódigo sería:

```
numericas: numero, raíz

inicio
mostrar por pantalla “introduce un numero”
leer numero
Inicio SI
SI numero>=0 ENTONCES: // >=0 significa mayor o igual que cero;
raiz=raiz_cuadrada(numero)
mostrar por pantalla “la raíz cuadrada es:”
 mostrar por pantalla raíz
 finSI
fin
```

En todos los lenguajes de programación **la condicional SI** se escribe de la siguiente forma.

if numero = 0 { órdenes que hará el programa si cumple la condición de que la variable numero sea igual a 0}

Es decir la palabra if seguida de la condición y seguidamente, entre corchetes, lo que se realizará si se cumple la condición.

Alternativas dobles (IF…….ELSE….)

¿Qué pasa si no cumple la condición puesta?. Pues si no le decimos nada, el programa seguirá a la siguiente orden de forma secuencial. Pero también podemos especificar que pasaría si no cumple la condición. Es el famoso trío "si ... entonces ... sino esto otro". Veamos como sería la estructura en todos los lenguajes:

if (condición) {se hace esto} else {si no cumple la condición se hace esto otro};

En el ejemplo anterior sería mucho mejor hacerlo con este tipo:

En pseudocódigo

```
numericas: numero, raíz
fin declaración de variables
inicio
Escribir 'introduce un numero'
Leer numero
InicioSI
SI numero >= 0 ENTONCES:
raiz = raiz_cuadrada(numero);
Escribir 'la raíz cuadrada es:' + raiz;
 SINO Escribir 'lo siento, no puedo calcular la raíz cuadrada de un numero negativo'
finSI
fin
```

Recuerda que si el número es menor de 0 sería negativo y no existen raices de números negativos.

Si te has fijado podemos poner Escribir "un texto" + variable (texto y a continuación aparecerá el valor de la variable en ese momento).

Cuando escribamos nuestro programa, en lugar de pseudocódigo, debemos poner la condición de la siguiente manera:

if numero >= 0 {raiz = raiz_cuadrada(numero); Escribir 'la raíz cuadrada es:' + raiz;} else { Escribir 'lo siento, no puedo calcular la raíz cuadrada de un numero negativo"}

Fíjate que podemos escribir todas las órdenes que queramos dentro de los corchetes siempre separadas por ;.

Cuando te encuentres con un programa real las órdenes dentro de un corchete verás que suelen ponerse de esta forma:

```
if numero >= 0 {
   raiz = raiz_cuadrada(numero);
  Escribir 'la raíz cuadrada es:' + raiz;
} else {
 Escribir 'lo siento, no puedo calcular la raíz cuadrada de un numero negativo"
}
```

Es lo mismo que antes, pero cuando tengamos que depurar (repasar) el programa visualmente nos será mas sencillo.

Alternativas múltiples o con varias condiciones

Es muy probable que tengamos la necesidad de incluir en nuestros programas alternativas con muchas opciones posibles.

variableOpciones= un valor a elegir, por ejemplo desde el teclado o desde una ventana que marque el usuario;

```
if (variableOpciones=0) {lo que corresponda};
if (variableOpciones=1) {lo que corresponda};
if (variableOpciones=2) {lo que corresponda};
```

Podemos poner tantas if como queramos.

También existe la posibilidad de que deban de cumplirse dos condiciones a la vez:

if (condición1 && condición2) {Se cumple esto}

También con else:
if (condición1 && condición2) {Se cumple esto} else {se cumple esto otro}

Los símbolos && significan "y", es decir si se cumple la condición1 y la condición2 a la vez (**las dos**).

Otro caso sería **si se cumple una cualquiera de las dos condiciones**:

if (condición1 | condición2) {Se cumple esto}

Como ves es el símbolo | (barra recta vertical del teclado = AltGr + 1)

Intenta hacer los siguientes ejercicios:

Sobre estructuras secuenciales

1. Escribe un algoritmo o pseudocódigo que calcule el área de un triángulo o de un rectángulo.

2. Escribe un algoritmo o pseudocódigo que calcule el precio de un artículo tras aplicarle un 16% de IVA.

Sobre estructuras selectivas

3. Diseña un esquema de menú de opciones, por ejemplo, un menú para seleccionar un libro a leer de entre 3 disponibles.

4. Escribe un algoritmo que lea tres números e imprima por pantalla el mayor de ellos.

Estructuras Repetitivas o Bucles DESDE o "FOR".

Estas estructuras son **instrucciones que se repiten formando un bucle** (algo que se repite una y otra vez).
A la variable que "lleva" la cuenta de las veces que el bucle se ha ejecutado, se le he llamado variable**contador**.

La estructuras FOR tienen la peculiaridad, que la variable contador está dentro del bucle y no hace falta asignarle el valor (ni definirla) fuera del bucle, y además, al llegar el programa al bucle siempre se realizarán las instrucciones que hay dentro del bucle, una cantidad de veces que nosotros fijemos.

Hay varias, pero esta que explicamos es la más utilizada.

Vamos a suponer que estamos pensando en un programa que deba **REPETIR** algunas veces una acción.
Un ejemplo más concreto. El ordenador se ha portado mal, y como castigo, le vamos a hacer imprimir por pantalla 3000 veces la frase "Prometo ser bueno".
¿Cómo lo hacemos? ¿Escribimos 3000 veces la instrucción pertinente?
¡Se supone que el castigo es para la máquina, no para uno mismo!

Veamos como sería el pseudocódigo:

```
inicio
   Inicio Bucle Desde
   desde i=1 hasta i<=3.000
    i=i+1;
   Escribir 'Prometo ser bueno';
   fin desde;
fin
```

Como vemos la variable i (llamada contador) no se define antes del bucle. Al entrar en el bucle i valdrá 1 (toma el valor inicial que le pongamos en el primer igual, en nuestro caso i= 1. Después le decimos hasta que valor de i se repetirá el bucle, en nuestro caso hasta que i valga menos o igual a 3000. Posteriormente ponemos cuanto aumenta la variable, en nuestro caso añadimos 1 al valor de la variable i; i = i + 1. Al final ponemos las órdenes que queramos que haga el programa cada vez que haga el bucle; en nuestro caso escribir en pantalla "Prometo ser bueno".

Como ves la primera vez que entra el programa en el bucle i vale 1, después i vale 2 (se le suma 1) y después escribe la fase. Antes de salir del bucle vuelve a evaluar la condición para ver si la sigue cumpliendo, si es así vuelve hacer el bucle entero. ¿Es así? Pues claro porque i=2 sigue siendo menor de 3000.

Pero ojo la segunda vez que hace el bucle i tomará el valor de 3, ya que le sumará 1 al valor que tenía, y recuerda que como ya hizo el bucle una vez ahora i = i + 1 será 3; ya que i = 2 +1.

Ves que cada vez que hace una vez el bucle el valor de i aumenta 1. Esto es lo que se llama "**el paso**". Podríamos hacer el bucle con paso 2 simplemente haciendo i = i +2.

Bueno siguiendo con el bucle, resulta que este bucle se repetirá hasta que i valga iguala o menor de 3000. Bueno pues en todas esas repeticiones el ordenador escribirá la frase: Prometo ser bueno. Castigo cumplido.

En lenguaje de programación real las estructuras For se forman:
for (i = 1 ; i<= 3000 ; i = i + 1)

A veces podrás ver esto:
for (i = 1 ; i<= 3000 ; i++)
i++ significa lo mismo que i = i +1; es especificar el paso 1 de i pero de otra forma, nada más

Recuerda en programación real se usa la palabra **for**, y no desde. Y normalmente la variable en lo bucles for se llama i.

¿Y si quisiéramos poner un paso decreciente? Es decir que el valor de i fuera disminuyendo cada vez que se repite el bucle. Pues muy sencillo poniendo i = i - 1.

Ejercicio: Programa que escriba los números del 1 al 10 con FOR
Con estos conocimientos básicos ya estas preparado para aprender a programar en cualquier lenguaje de programación de alto nivel. Ahora tu elijes el que quieres aprender, veras que con estos conocimientos te será Muy Sencillo, solo aprender un poco de vocabulario del lenguaje de programación en ingles. ¡¡¡Suerte!!!.

PREGUNTAS DE CONTROL

1. ¿Qué es el seudocódigo y cuál es el principal problema de los hispanos al momento de programar?
2. ¿Cuáles son las dos palabras que se utilizan para escribir el seudocódigo entre ellas?
3. ¿Cuáles son las órdenes que más se utilizan en el seudocódigo?
4. ¿Cuál es la norma para escribir comentarios en un programa?
5. ¿Qué es una variable y como se nombran?
6. ¿Cuáles son los tipos de variables, según su valor, explique su reglas y de ejemplo de cada uno de ellas?
7. ¿Cuáles son los resultados de las siguientes operaciones con variables? Teniendo en cuenta que:
   Va = 4;
   Vb = 5;
   Vc = Va+Vb
   A) Vc – Va B) Vb*Va C) (Va+Vb)-Vc D) (Vb*Va)+Vc
8. ¿Cuál es la principal diferencia entre variables locales y globales?
9. Construya un mapa conceptual con las estructura de control
10. Realice un ejemplo con cada una de las estructuras de control y sus subdivisiones