Manual de usuario

TypeWise

Este lenguaje de programación es un lenguaje interpretado, esto quiere decir que no compila el código, sino que lo ejecuta directamente.

Sintaxis del lenguaje TypeWise:

Declaración de variables:

Declaración 1:

Para la declaración de variables se deberá colocar el tipo de valor que almacenará dicha variable, seguido del valor

```
void suma(){
  int a= 0;
}
```

Declaración 2:

Si lo que desea es asignar un valor posteriormente en la ejecución del programa puede declararla sin asignar ningún valor. Por ejemplo:

```
void suma(){
  int a;
}
```

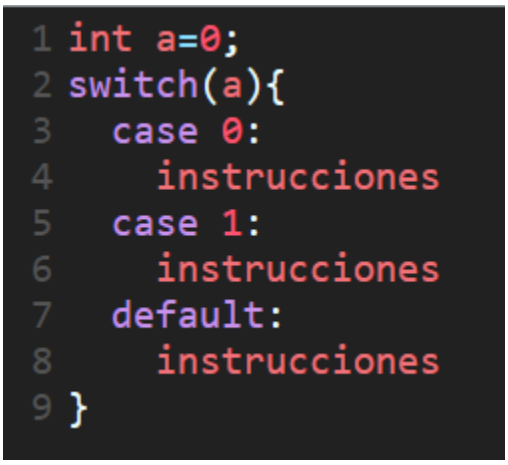
Sentencias de control:

El lenguaje permite 2 sentencias de control, estas son la sentencia if-else y la sentencia switch-case. La sintaxis de la sentencia if-else es la siguiente

```
if(condicion1){
  intrucciones
}else if(condicion2){
  instrucciones
}else{
  instrucciones
}
```

Sentencia Switch-case:

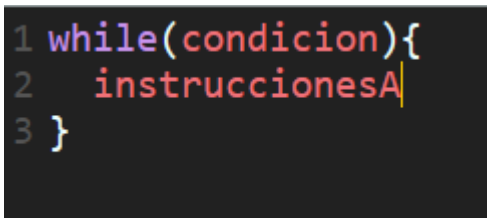
Esta sentencia permite comparar múltiples condiciones, abreviando la sentencia if



```
int a=0;
switch(a){
  case 0:
    instrucciones
  case 1:
    instrucciones
  default:
    instrucciones
}
```

Así mismo el lenguaje TypeWise permite ejecutar 3 sentencias cíclicas, estos son el ciclo while, el ciclo for y el ciclo do-while.

Bucle while:

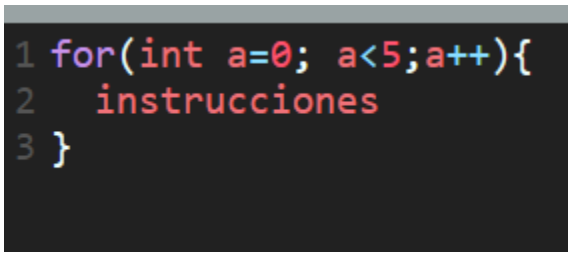


```
while(condicion){
  instruccionesA
}
```

El bucle while permite ejecutar un bloque de instrucciones si se cumple una condición. Este ciclo ejecuta 0 o muchas veces el bloque de instrucciones.

Bucle For:

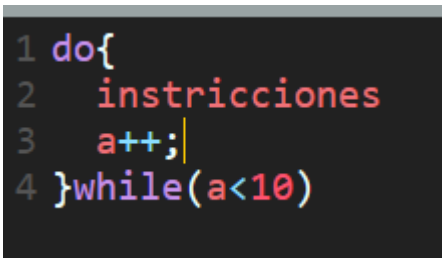
Este bucle al igual que el bucle anterior permite ejecutar un bloque de sentencias si se cumple una condición, pero a diferencia del anterior cuenta con un contador, que aumenta o disminuye según se decida automáticamente al finalizar el bloque de instrucciones.



```
for(int a=0; a<5;a++){
  instrucciones
}
```

Bucle Do-While:

Este bucle es parecido al bucle while, la única diferencia radica en que el bloque de instrucciones se ejecutará al menos una vez.



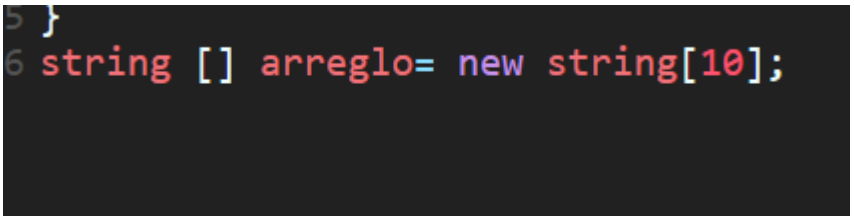
El lenguaje TypeWise permite la creación de estructuras de datos, tales como listas y arreglos.

Arreglos:

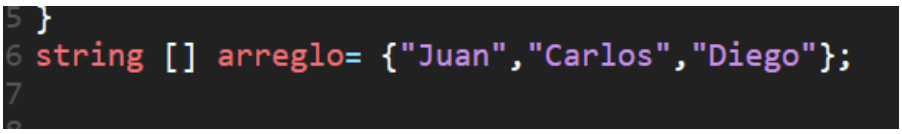
Los arreglos en el lenguaje TypeWise son estáticos y no permiten modificar su tamaño. Hay 2 formas de declarar arreglos:

1. Especificar su tamaño
2. Ingresar directamente los valores

Especificar su tamaño:



Ingresar los valores directamente:



Listas:

Las listas a diferencia de los arreglos son dinámicos por lo que permiten almacenar muchos valores del mismo tipo. La declaración de las listas es de la siguiente manera.

```
9
10 list <double> list1= new list<double>
```

Agregar valores a las listas:

Para agregar valores a las listas se debe realizar con la palabra reservada add

```
11 list1.add(1.12);
12 list1.add(0.24);
```

Creación de funciones:

El lenguaje TypeWise también permite el uso de funciones con y sin parámetros.

La sintaxis para la creación de una función sin parámetros es la siguiente:

```
1 void saludar(){
2   print("Hola");
3 }
```

Se debe colorar el tipo de retorno de la función seguido del nombre y por ultimo el bloque de instrucciones.

Creación de funciones con parámetros:

```
1 void suma(int a,int b){
2   print(a+b);
3 }
```

Para construir una función que recibe parámetros dentro de los paréntesis se deben color tanto el tipo de parámetro que se recibe, así como el nombre de la variable.

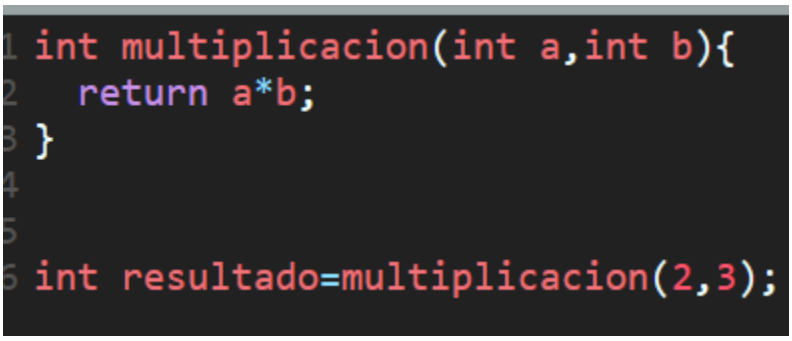
Funciones con retorno:

Son las funciones que no son de tipo void.

Estas funciones pueden retornar una operación, una comparación o incluso una llamada a función.

Llamadas a funciones:

Se pueden realizar 2 tipos de llamadas a funciones, estos son llamadas a funciones declaradas que reciben parámetros y las que no reciben.



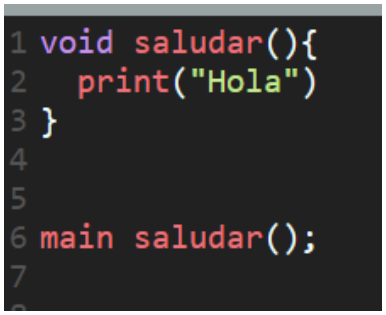
```
int multiplicacion(int a,int b){
  return a*b;
}


int resultado=multiplicacion(2,3);
```

En este caso la función recibe 2 parámetros de tipo entero por lo que la llamada se envíen los 2 parámetros correspondientes.

Función Main:

Esta función llama a la función con la que se iniciara la ejecución del programa, si esta no encontrada, el programa no se ejecutará



```
void saludar(){
  print("Hola")
}


main saludar();
```