Sintaxis básica

Durante la materia vamos a usar **pseudocódigo** o **C** para resolver problemas en exámenes y en clase. Nos va a interesar el **orden temporal** de las funciones que usemos para resolver de la forma más eficiente lo pedido.

Tipos de datos

Para declarar una variable vamos a indicar el tipo precediendo al nombre de la variable.

```
<tipo> nombre = valor
```

Los tipos disponibles son:

Numérico: Un número con o sin decimales

String: Una cadena de caracteres. Se puede operar con ella como en Python.

Booleano

Tipos de datos abstractos: Otros tipos de datos

Funciones

Para definir una función escribimos una firma donde indicamos que tipo devuelve la función y que tipos recibe, como lo haríamos en C.

```
numeric producto(numeric a, numeric b){
  return a*b
}
```

Podemos asumir que tenemos una bateria de funciones disponibles:

Nombre	Descripción	Orden temporal
sort	Ordena un arreglo/string	O(n log(n))
reverse	Invierte un arreglo/string	O(n)
max/min	Obtiene el máximo o mínimo de un arreglo	O(n)

Puede asumir que tiene disponible tambien funciones para operaciones matemáticas (*sqrt*, *cos*, ...) y preguntar en caso de necesitar otra no listada.

Sintaxis básica

Condición IF

Las condiciones IF tienen el siguiente formato:

```
if (promedio > 4){
  return TRUE
}
```

Ciclos

Los ciclos definidos pueden expresarse de dos formas:

```
for i in [1,2,3,4]{
  return TRUE
}
```

```
for i in 1..4{
  return TRUE
}
```

Incluyendo en la iteración al primer y último elemento.

Los ciclos indefinidos se expresan de la siguiente forma:

```
while (continuar){
   ...
}
```

Ejemplo

Se pide una función que determine si un numero es primo.

```
boolean es_primo(numeric numero) {
    for i in 2..floor(sqrt(numero)) {
        if (numero%i == 0) {
            return FALSE
        }
    }
    return TRUE
}
```

Tipos de datos abstractos

Listas

```
lista crear_lista()

insertar_ultimo(lista lista, elemento)

insertar_primero(lista lista, elemento)

.. extraer_primero(lista lista)

.. extraer_primero(lista lista)

numeric cantidad(lista lista)

Colas
```

```
cola crear_cola()
encolar(cola cola, elemento)
.. desencolar(cola cola)

numeric cantidad(cola cola)
```

Pilas

```
pila crear_pila()

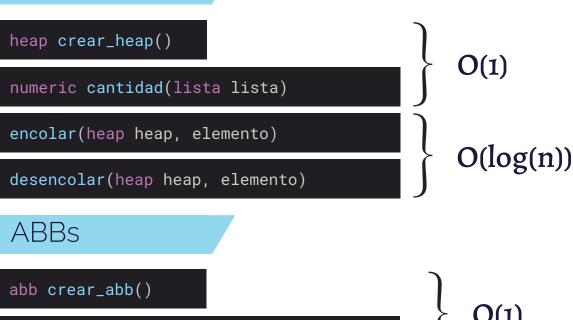
apilar(pila pila, elemento)

.. desapilar(pila pila)

numeric cantidad(pila pila)
```

Tipos de datos abstractos

Heaps



guardar(abb arbol, clave, elemento)

borrar(abb arbol, clave)

numeric cantidad(abb arbol)

.. obtener(abb arbol, clave)

boolean pertenece(abb arbol, clave)

O(1)

O(log(n))

Hash

hash crear_hash()		
numeric cantidad(hash hash)		
guardar(hash hash, clave, elemento)		
borrar(hash hash, clave)		
obtener(hash hash, clave)		
boolean pertenece(hash hash, clave)		

O(1)