

# PROGRAMACIÓN ESTRUCTURADA

## Elementos de Gobstones

**Tablero:** Grilla rectangular dividida en pequeñas Secciones (celdas), También de forma Vertical (columnas) y horizontal (filas)

**Celda:** Ubicación donde puede haber **bolitas** de Colores Negras, **azules**, **rojas**, Verdes. No hay nº máximo de bolitas y pueden combinarse de cualquier manera.

- Las celdas se identifican indicando Columna (de izquierda a derecha) y Fila (de abajo hacia arriba).
- La celda de otro color y remarcada es la **celda actual**, donde se encuentra posicionado el cabezal.

**cabezal:** "máquina" capaz de manipular las bolitas que se encuentran bajo su posición. (poner/sacar) y puede moverse de una celda a otra.

Encargado de ejecutar las acciones que describimos en un programa.

**Comandos (Bloques azules):** Descripciones de acciones. Acciones que el cabezal podrá realizar.

**Expresiones (Bloques verdes):** Descripciones de Información. Información específica útil para que el cabezal ejecute bien el argumento de un comando

## PROGRAMA

Texto que describe la solución a un problema computacional y debe poder ser ejecutado por una máquina.

La descripción está dada por comandos y expresiones.

El programa que escribimos debe poder ser ejecutado en diferido, tantas veces como se desee, siempre de la misma forma y entendido por cualquier máquina.

La solución debe ser metódica y reproducible.

Un problema computacional debe ser expresado como transformaciones de estado o de información. En Gobstone el Estado se representa mediante el Tablero con la posición del cabezal y cantidad/Color de bolitas por celda.

Los programas son equivalentes cuando solucionan el mismo problema

## Lenguajes de programación

Permiten la comunicación entre humanos y Computadoras, determinan la forma de describir nuestra solución ya que cada lenguaje determina que símbolos son válidos y qué reglas debemos seguir, esto es la **Sintaxis**. En cambio lo **Semántico** se refiere a qué ideas estamos transmitiendo a la hora de codear y deberían ser similares en todos los lenguajes

## Comandos primitivos en Gobstones

poner - sacar - mover - ir al borde

## Expresiones literales

Colores: Azul - Negro - Rojo - Verde

Direcciones: Norte - Sur - Este - Oeste

## Palabras claves (indicativo a la máquina particular)

program: punto donde comienza el programa

procedure: punto donde comienza un procedimiento

## CONVENCIONES

Gobstones utiliza CamelCase, consiste en poner en mayúscula la primer letra de cada palabra. CamelCase es una convención de estilo para entender el código.

La indentación es el tabulado del código y sirve para identificar visualmente donde comienza un cuerpo y donde termina. Hay que mantener el mismo estilo de indentación durante todo el código de un programa.

Un comentario es una parte del texto que no es contemplado por la máquina, funciona para transmitir información relevante entendido por personas como el autor del código, cuando, descripción del programa, etc.

## CONTRATOS

Es parte de la documentación del programa, información adicional para poder usarlos correctamente.


- Indica que hace el programa. **Propósito**
- Indica cuando funciona correctamente y cuando no. **Precond.**
- Ayuda a pensar el problema antes de intentar solucionarlo
- Sirve para poder reutilizar código y hacerlo más mantenible


**Propósito:** Dejar en claro cual es la Transformación del Tablero al ejecutar el código. Teniendo en cuenta las bolitas (cantidad y color) y la posición del cabezal (donde realiza la acción y donde termina).

Solo indicar si el cabezal termina en otro lado y no si termina donde comenzó

**Precondiciones:** restricción sobre el estado inicial, planteada como preguntas deberían ser respondidas por si o no.

### - Comandos parciales

**Mover** - Debe haber al menos una celda en la dirección que se da como argumento  cant mover

**Sacar** - Debe haber al menos una bolita del color que se da como argumento en la celda actual  cant bolitas

### - Comandos Totales

**Poner** - Siempre Funciona

**IrAlBorde** " "

- Un programa que usa comandos parciales podría tener precondiciones mientras que uno con comandos totales no

**Precondición débil** es la que indica lo mínimo y necesario para que el programa funcione correctamente, es la que buscamos siempre.

**Precondición Fuerte** sobredimensiona los requerimientos necesarios para funcionar

**EN LA CELDA ACTUAL**



## PROCEDIMIENTOS

- Mecanismo que permite definir comandos nuevos, tiene 2 partes:
  - + Definición de procedimiento: con la cabeza y cuerpo del procedimiento le damos un nombre y describimos, mediante acciones, que significa este nuevo comando
  - + Uso del procedimiento, se coloca en el programa o algún otro procedure para llamarlo o invocarlo
- Sirve para reutilizar código
- Su nombre debe comenzar con verbo infinitivo, en mayúscula y seguir camelCase
- Se usan letras, números y guión bajo
- Tienen contratos propios: sus propósitos son independientes del programa/procedimiento en donde se invocan. Sus precondiciones deben ser contempladas en el programa o procedimiento en donde se invocan.

## REPRESENTACIÓN DE PROBLEMAS

Es necesario dividir los problemas en niveles

1º Resolución de problema

2º Representación del funcionamiento

3º Representación básica

Los procedimientos también nos permiten abstraernos de la representación (3º nivel) aclarando en observaciones las representaciones necesarias. Para luego centrarnos en crear procedimientos que hablen de la representación del problema (2º nivel).

Esto aporta legibilidad al programa en general

## Pasos a seguir para parametrizar procedimientos

- 1 Identificar procedimientos parecidos que solo difieran en un dato
- 2 Dejar un agujero en los lugares en donde se usa el dato que se cambia
- 3 Dejar un único procedimiento ya que todos terminan siendo iguales, usando notación me-fix
- 4 Agregar un parámetro por cada agujero que tenemos
- 5 Cambiamos las invocaciones anteriores por las nuevas