- Clase referente al 19/5/2017

Lo que vamos a ver hoy, es basicamente como interpretar los enunciados de los algoritmos. Las cosas que debemos tener en cuenta al inicio de implementar el codigo y tambien, ver un poco la sintaxis que estuvimos ya hablando con anterioridad.

- El siguiente enunciado ya lo han resuelto en el examen, pero vamos a leerlo, interpretarlo y resolverlo:

    - Ingresar la altura de una persona e imprimir si es Baja, Normal o Alta, de acuerdo con las siguientes condiciones:
        A. Si la altura es menor a 1.60, es una persona baja.
        B. Si la altura esta comprendida entre 1.60 y 1.75 (ambos inclusive), es una persona Normal.
        C. Si la altura supera a la medida de 1.75, es una persona Alta.

- Lo primero que debemos analizar son las Entradas - Procesos - Salidas.

    En este caso entendemos como Entradas - Proceso - Salida a lo siguiente:

    - Entradas: La altura de la persona.
    - Proceso: Comparaciones entre la altura ingresada y las que propone el punto A,B y C.
    - Salidas: Los print que nos indicaran si la persona es baja, normal o alta.

- Lo segundo que debemos analizar son los tipos de datos que van a ingresar:

    Analizando los datos de entrada sabemos que la altura se mide en un numero real con dos decimales (Ej. 1,70), entonces debemos conocer que el tipo de variable que se corresponde a este. En este caso, sabemos que va a ser un numero **float** (aunque para tener dos decimales podemos usar el tipo **double**).

- Tipo de datos: **float**

A la hora de crear el algoritmo podemos usar varias sintaxis, siempre usando la identacion adecuado. Aca les dejo este problema resuelto con la sintaxis que yo utilizo:

```
Begin
    float altura;
    print("Ingrese la altura: ");
    read(altura);
    if (altura < 1.60) {
        print("La persona es baja");
    }
    else {
        if ((altura >= 1.60) && (altura <= 1.75)) {
            print("La persona es normal");
        }
        else {
            print("La persona es alta");
        }
    }
End
```

Podemos observar lo siguiente:

    - Siempre que terminemos una instruccion (nunca las estructuras de control), la finalizamos con punto y coma (;). Esto le dice a la computadora que ahi termina la instruccion, ya que la computadora no discrimina por espacios ni salto de lineas.
    - Tenemos que saber indentar, son los espacios que dejamos en los bloques de codigo (TAB), se tiene que seguir una linea imaginaria de una sentencia a la otra, lo mismo con las estructuras de control.
    - Las estructuras de control las iniciamos con { y las cerramos con }, esto forma parte de mi manera de realizar algoritmos y esta incluido en la mayoria de los lenguajes, pueden optar por esta manera o tambien por un End son los corchetes, yo les recomiendo adoptar esta opcion para que todos realicemos algoritmos de la misma manera.
    - Siempre los String van a estar entre comillas dobles (" ") y los **char** en comillas simples (' ') (Esto depende del lenguaje, ya que en algunos es indistinto) aqui los vamos

a diferenciar. Ej, un String "Nombre" y un **char** 'A'.
    - Siempre la asignacion de variables se van a hacer con el signo igual (=). Ej contador = 0;.

Espero que les haya quedado claro esta parte.

Ahora pasemos a la estructura FOR que estuvimos viendo hoy en clase y que tienen en el material teorico la explicacion mas completa.

La estructura **for** tiene 3 partes:

    - Inicializacion
    - Condicion de Corte
    - Incremento

Ej: **for** (Integer i = 0; i < 10; i++) {
        ...
}

Si dividimos las partes "Integer i = 0;" esta es la inicializacion de la variable. "i < 10; " es la condicion de corte y "i++" es el incremento en 1 de la variable "i" que estamos usando en el la inicializacion.

Veamos un ejemplo:

    - Ingresar 15 valores, e imprimir:
        A. Cuantos fueron pares.
        B. Cuantos fueron impares.
        C. Cuantos terminados en cero.

- Siempre deben tener en cuenta que la estructura **for** se utiliza SIEMPRE que sabemos cuantas veces tenemos que repetir algo. Cuando no sabemos la condicion de corte, debemos usar un **while** que veremos la proxima clase.

- Razonando el enunciado entendemos que vamos a REPETIR 15 VECES algo, saquemos las Entradas - Procesos - Salidas.

        - Entradas: 15 numeros enteros (Ya que vamos a evaluar par e impar, aunque el problema no lo especifique).
        - Procesos: Contar cuantos fueron pares, impares y ceros.
        - Salidas: Imprimir el resultado **final** de cada contador.

```
Begin
    Integer num, par, impar, ceros;
    num, par, impar = 0;
    for (Integer i = 0; i < 15; i++) {
        print("Ingrese un numero: ");
        read(num);
        if (num % 2 == 0) {
            par++;
            if (num % 10 == 0) {
                ceros++;
            }
        }
        else {
            impares++;
        }
    }
    print("La cantidad de numeros pares, impares y terminados con ceros es:" + par + ", " +
 impar + ", " + ceros);
End
```

Analicemos:

    - Razonamiento principal:

        - Nosotros conocemos que los numeros terminados en 0 siempre van a ser pares.

- *Para que el algoritmo sea lo mas correcto posible, debemos primero poner la condicion de paridad, y dentro de la de paridad, la condicion de evaluar si termina en cero.*
- *Si no se cumple el primer If, sale por el Else y quiere decir que es impar y no evaluamos si termina en cero porque no es necesario en ese caso.*
- *Esto es un algoritmo eficiente, hay otras maneras de resolverlo pero deben usar siempre el mejor razonamiento. Nunca razonen con lo mas simple, siempre busquen lo mas eficiente para la computadora. Tener en cuenta que la eficiencia no esta solo en la programacion, sino en muchos otros ambitos. Si nos acostumbramos desde ahora a hablar de eficiencia, todos los ejercicios van a ser sumamente mas faciles, porque el razonamiento de los ejercicios simples, es escalable a los ejercicios mas complejos donde vamos a necesitar enfocarnos en otras cosas ademas de la eficiencia.*

- *Siempre los contadores van a inicializarse en* 0, *ya que vamos a usar los incrementos* ++, *quiere decir que al valor que tiene el contador se le va a sumar uno. Seria como* "par = par + 1;", *siempre debe tener algo esa variable antes de incrementarla.*
- *Debemos tener en cuenta* (Integer *i* = 0; *i* < 15; *i*++), *se puede ver que* "i" *inicia en* 0 *y el signo que usamos es el* "<" *en la condicion, la iteracion va a ser de* 0 *a* 14 (15 *veces*). *Tambien podemos usar* (Integer *i* = 1; *i* <= 15; *i*++), *donde* "i" *inicializada en* 1 *va a iterar* 15 *veces* (1 *a* 15). *Esto debemos tenerlo en cuenta ya que podemos usar el valor que tenga* "i" *en todo momento dentro del* **for**, *ya que es una variable privada de esa estructura o bloque.*
- *Tener en cuenta que* "num % 10" *nos dara como resultado el ultimo digito del numero que ingresemos* (*siempre y cuanto sea entero*) *quiere decir que si hacemos* "1234 % 10" *nos va a devolver* "4". *Otra cosa a tener en cuenta es que si en vez de hacer una division entera* "%" *hacemos una division comun* "/" *nos va a devolver todos lo digitos menos el ultimo, es decir que si hacemos* "1234 / 10" *nos va a devolver* "123".

Espero que les haya quedado claro la clase de hoy a los que no pudieron ir. Les dejo ejercicios para realizar.