**Eficiencia:** tiempo y espacio  
Clase 2/5 continuación de 25/4

**Eficiencia:**

* “O” tal que
* O(n): lineal
* O(1): constante

Las operaciones, acciones y sentencias, llevan tiempo.

“>>”: extracción

“<<”: inserción

Ej:

1.

2 objetos

4 operaciones

O(n) = 4

std::cin >> a;  
std::cin >> b;  
std::cout << a+b;

2.

int a;  
int b;  
cin >> a;  
cin >> b;  
b = a+b;  
cout << b;

2 objetos

5 operaciones

O(n) = 5

La entrada y salida de datos es el proceso mas lento que hay. La adición o sustracción es el proceso mas rápido.  
La asignación es rápida también, por ser resuelta en el microprocesador, en cambio con la inserción y la extracción se tiene que alejar físicamente hacia el teclado y la pantalla.

O(n)= en función de la cantidad de elementos a procesar  
O(1)= no importa la cantidad de elementos a procesar

Siempre se busca que la eficiencia sea lineal o menor, la peor situación es una de tipo O(n2) (cuadrática).

**Tipo de dato:**

* **Conjunto de operaciones sobre esos valores.**
* Ejemplo:
  + Un nombre, donde los valores es la secuencia de letras. Tal que sean no vacías y finitas (estas son restricciones sobre los valores). La operación posible puede ser concatenación (*José* *María*).
* Los tipos de datos los creamos definiendo los conjuntos de valores y las operaciones posibles sobre estos

Representación

Por ejemplo, si queremos representar en c++ un punto del plano:

struct Punto {  
 double x, y;  
};

Funciones

double f(double); <- Prototipo / Declaración

double f(double x) { return 2\*x+1; } <- Definición

* cassert: sirve para probar las cosas.

Para funciones seguimos el estilo (ver el cpp):

* Declarar
* Invocar
* Definir

Especificar: Definir el conjunto de salida de llegada y la definición

Convención durante la programación:

* Si algo es **global**, entonces es en **mayúscula**.
* Si algo es **local**, entonces es en **minúscula**.

Operador ternario:

A ? B : C

Si A es verdadero, retorno B. Si A es falso, retorno C

Este estilo sirve para cuando solo buscamos un valor con 2 condiciones. De esta manera no es necesario invocar a una función condicional *if.*