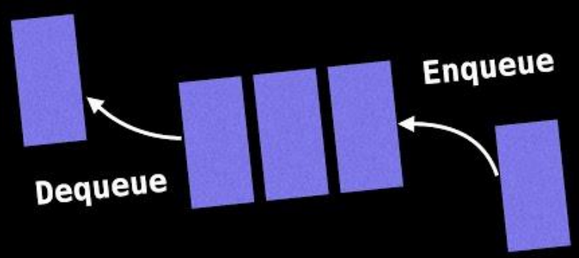
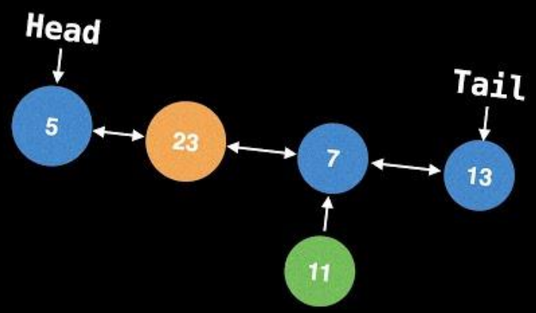
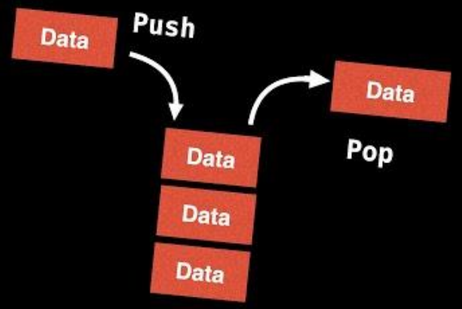
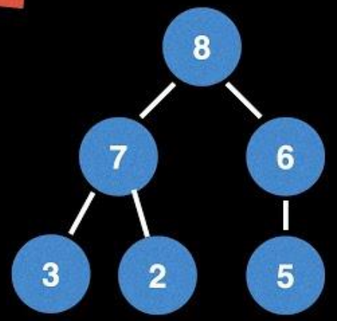
Welcome to

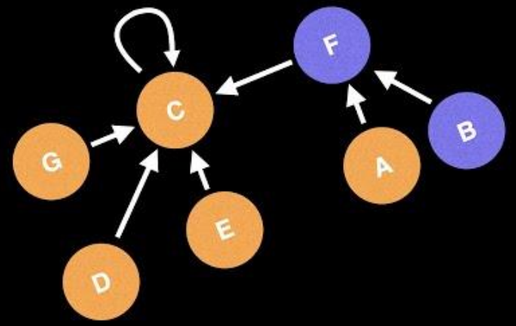




Data Structures







CS2100

Heider Sanchez E., PhD
hsanchez@utec.edu.pe

Qué veremos en este curso...

- Algoritmos para resolver problemas de manera eficiente
- Estructuras de datos para almacenar y organizar datos eficientemente
- Análisis de requerimientos para uso adecuado de distintas estructuras de datos

Qué NO veremos en este curso...

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

- POO se asume como algo estudiado en cursos previos.
- No se enseñará a compilar en C++, ni a usar algún IDE específico.
- El manejo de Punteros en C++ se asume como algo aprendido en un nivel intermedio-avanzado.
- La STL y estructuras básicas se entienden como ya vistas en cursos previos.

  Tutorial rápido de STL C++

Lista de temas

Semana	Tema
1	Punteros, Arrays
2	Ordenamiento, Listas
3	Listas, Iteradores
4	Tabla Hash
5	Árbol Binario de Búsqueda
6	Heaps
7	Disjoin Set
8	Matrices Esparza
9	Grafos
10	Búsqueda en Grafos 1
11	Búsqueda en Grafos 2
12	Árboles AVL
13	Árboles B / B+
14	Tries
15	Optimización

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Laboratorio

1. Si el entregable no compila será calificado sobre 11
2. Tratar de evitar warnings en los proyectos
3. Tratar que pasen todos los tests de prueba.
4. No olviden subir su código a master

Sistema de evaluación

EVALUACIÓN	TEORÍA (T)	LABORATORIO (L)
* La ponderación de la evaluación se hará si ambas partes están aprobadas	Examen **E1** (20%) Examen **E2** (20%)	Evaluación Continua **C1** (20%) Evaluación Continua **C2** (20%) Proyecto **P1** (10%) Proyecto **P2** (10%)
	40%	60%
	100%	

Evaluación Continua:

- Tareas en Github Classroom (grupales/individuales)
- Ejercicios en clase (individuales)

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Herramientas

- **Compilador: C++ 17**
  (GCC, Cygwin, MinGW-w64, MSYS2)
- **IDE: Visual Code, CLion**
- **Git + Git Flow + Fork**
- **Karma Git Commits**
  a. ***http://karma-runner.github.io/4.0/dev/git-commit-msg.html***

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Comunicación

Discord:

SCAN ME

Tu NickName debe ser tu nombre y apellido

Revisar Git / GitHub

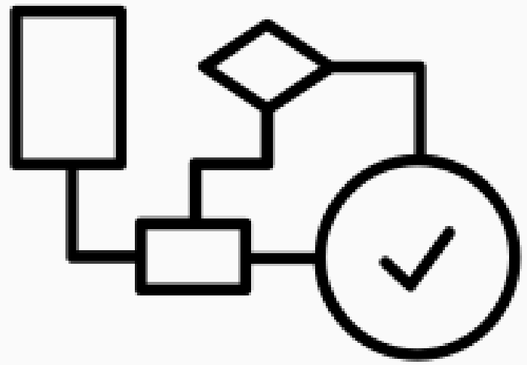
1. Crear una cuenta en github (https://github.com/)
2. Revisar: clone/add/commit/push/pull
3. Hacer push de un main.cpp (Hello world!)
4. Revisar branch/checkout/merge
5. Crear una rama develop, modificar el main.cpp
6. Hacer un commit y merge con master

¿Qué es un algoritmo?

➔ Es un conjunto ordenado de pasos o instrucciones, los cuales son realizados para resolver un problema.

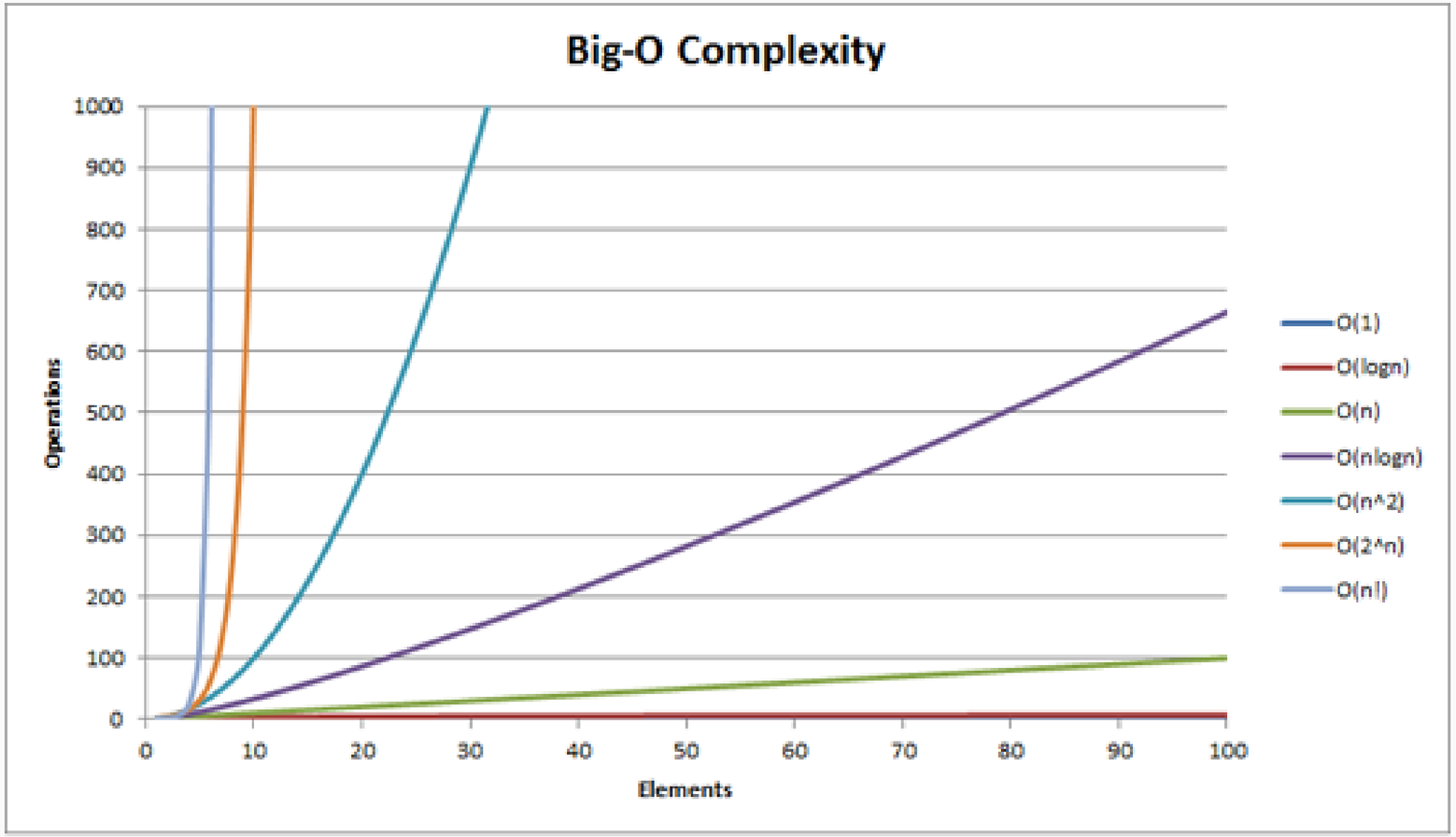
➔ Un algoritmo siempre debe producir un resultado, por ello se debe hacer de forma racional y con un objetivo en mente.

E.g. Una receta de cocina, la manera en la que se muestran los resultados de Google, etc.



Qué es la notación big O?

Provee un límite superior a la tasa de crecimiento de una función



Qué es la notación big O?

- T = a = **O(1)**
- T = an + b = **O(n)**
- T = $an^2 + bn + c$ = **$O(n^2)$**

$\log n$	n	$n \log n$	n^2	n^3	2^n
0	1	0	1	1	2
1	2	2	4	8	4
2	4	8	16	64	16
3	8	24	64	512	256
4	16	64	256	4096	65,536
5	32	160	1024	32,768	4,294,967,296

En qué casos sería O(log n)?

Ordenamiento

Qué ejemplos de ordenamiento se les ocurren?
Ordenar a personas en base a la edad, ordenar entradas de un blog en base a fecha, etc.

Qué algoritmos de ordenamiento conocen?
Bubble sort, selection sort, insert sort, quick sort, merge sort, heap sort, counting sort, etc.

En qué nos basamos para elegir un algoritmo de ordenamiento para un proyecto?
Algunos son más rápidos pero usan muchos recursos, otros lentos pero ligeros y usan pocos recursos.
Esto los hace un gran caso de estudio para comparar.

Vamos a recordar un poco… Punteros

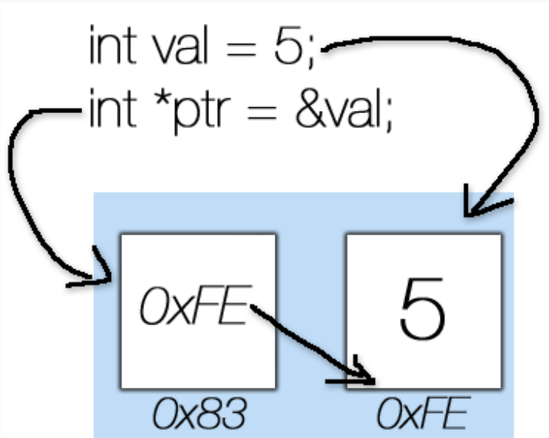
Qué es un puntero?
Un puntero es una variable la cual almacena la dirección de otra variable. Para declarar un puntero se utiliza la siguiente nomenclatura:
*<tipo> *<nombre>;*

Dada la tabla de la derecha, cuánto espacio ocupa un puntero en memoria?
Un puntero ocupa 4 bytes en arquitecturas de 32 bits y 8 bytes en arquitecturas de 64 bits. No depende del tipo de dato.

Type Name	32-bit Size	64-bit Size
`char`	1 byte	1 byte
`short`	2 bytes	2 bytes
`int`	4 bytes	4 bytes
`long`	4 bytes	8 bytes
`long long`	8 bytes	8 bytes

Punteros

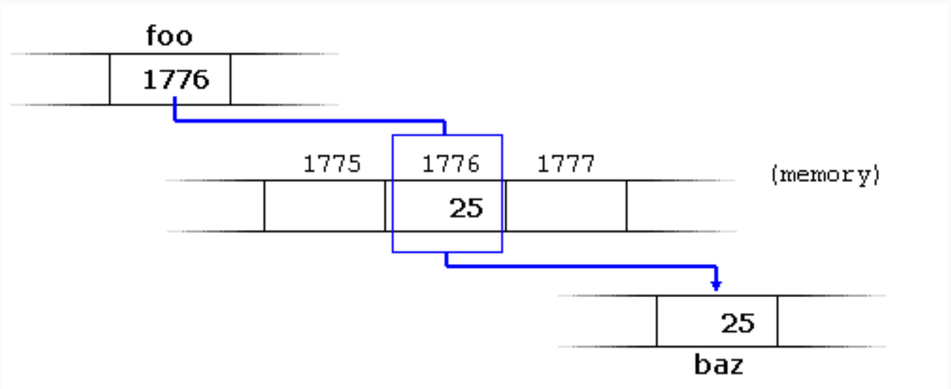
Una variable que almacena la dirección de otra variable es un puntero. Por tanto se suele decir que un puntero **“apunta a”** la variable la cual su dirección está almacenada



Operadores importantes:

Dereference (*):

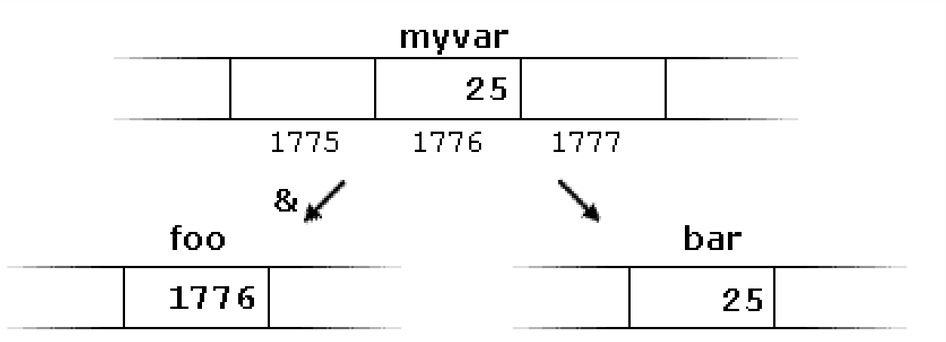
a. Se utiliza para definir al puntero, y va de lado de la variable. E.g. int *a, *b, *c;
b. A su vez, permite acceder al contenido a la cual apunta. E.g. int d = *a;



Punteros

Address-of (&):

a. Obtiene la dirección de una variable de cualquier tipo. E.g. int *pointer = &variable;



Por qué el tipo de dato debe ser conocido al momento de definir un puntero?
Se debe a una de las características de *Dereference*, ya que podemos obtener el valor al cual se apunta.

Recuerden que la verdadera dirección solo se puede conocer en tiempo de ejecución

Punteros

Cuando un array es declarado, se separa suficiente memoria para almacenar todos los elementos, dado:

```
short arr[5] = {1, 2, 4, 8, 16};
cout << arr << endl;
```

Qué imprimirá el código anterior?
1000, ya que *arr = &arr[0]*

y esto?

```
cout << *arr + 2 << endl;
// Sería 3!
```