# Heap

Se incluye en el sitio de descargas el archivo `heap.h` correspondiente al ejercicio de la cola de prioridad.

El trabajo que deben entregar de **forma grupal** es el tipo de dato abstracto Cola de Prioridad, utilizando un Heap.

## Primitivas del heap

```c
typedef struct heap heap_t;
typedef int (*cmp_func_t) (const void *a, const void *b);

heap_t *heap_crear(cmp_func_t cmp);
heap_t *heap_crear_arr(void *arreglo[], size_t n, cmp_func_t cmp);
void heap_destruir(heap_t *heap, void destruir_elemento(void *e));

size_t heap_cantidad(const heap_t *heap);
bool heap_esta_vacio(const heap_t *heap);

bool heap_encolar(heap_t *heap, void *elem);
void *heap_ver_max(const heap_t *heap);
void *heap_desencolar(heap_t *heap);
```

Además, deben implementar el ordenamiento heapsort, sobre un arreglo de punteros genéricos; y las pruebas unitarias de todas las primitivas implementadas.

```c
void heap_sort(void *elementos[], size_t cant, cmp_func_t cmp);
```

La función de comparación (de tipo `cmp_func_t`) debe recibir dos punteros del tipo de dato utilizado en el heap, y debe devolver:

- menor a 0 si a < b
- 0 si a == b
- mayor a 0 si a > b

Como siempre, deben subir el código completo a la página de entregas de la materia y también entregarlo impreso con nombre y padrón de ambos integrantes, si su corrector así lo requiere.

**No olviden revisar las preguntas frecuentes del heap**

---

# Bibliografia recomendada


- Weiss, Mark Allen, “Data Structures and Algorithm Analysis”: *Chapter 6: Priority Queues (Heaps)*.
- Cormen, Thomas H. “Introduction to Algorithms”: *6.5. Priority queues*, *6.1. Heaps*, *6.2. Maintaining the heap property*.