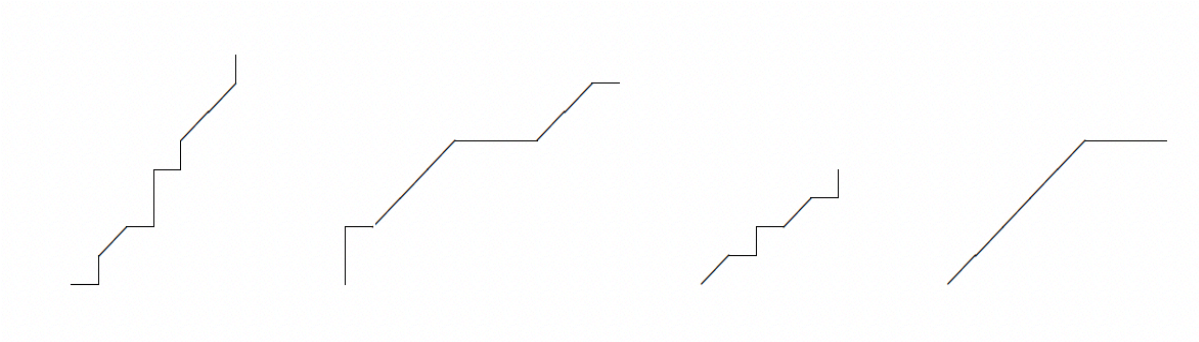
# Algoritmos y Estructuras de Datos II

## Parcial 30-05: Tema B - TAD: Caminata

El alumno deberá implementar el tipo abstracto de datos caminata (ver especificación abajo) en el lenguaje de programación C, utilizando la técnica de ocultamiento de información visto en el taller de la materia.

Se define el TAD caminata, que es el tipo de las caminatas. Intuitivamente, una caminata es una secuencia de pasos hacia arriba (N = norte), hacia la derecha (E = este) o en diagonal (NE = noreste). A continuación, representaciones gráficas de algunas caminatas:



El tipo abstracto tiene 4 constructores: uno llamado V para crear la caminata vacía, otro llamado N que corresponde a agregar un paso hacia el norte al final de una caminata, otro llamado E que corresponde a agregar un paso hacia el este al final de una caminata, y finalmente, uno llamado NE que corresponde a agregar un paso en diagonal al final de una caminata.

El TAD posee además operaciones para calcular la longitud o número de pasos de una caminata, su ancho y su altura. Posee además una operación que extiende un camino con un cierto número de pasos hacia el este y un cierto número de pasos hacia el norte.

Las operaciones del TAD se listan a continuación:

| Función | Descripción |
|---|---|
| `walk_t walk_empty(void)` | Crea una nueva caminata vacía |
| `walk_t walk_north(_walk_t c)` | Agrega un paso al norte a una caminata c |
| `walk_t walk_east(_walk_t c)` | Agrega un paso al este a una caminata c |
| `walk_t walk_northeast(_walk_t c)` | Agrega un paso diagonal hacia el noreste |
| `unsigned int walk_length(_walk_t c)` | Calcula la longitud de una caminata |
| `unsigned int walk_height(_walk_t c)` | Calcula la altura de una caminata |
| `unsigned int walk_width(_walk_t c)` | Calcula el ancho de una caminata |
| `walk_t walk_extend(_walk_t c, unsigned int east, unsigned int north)` | Extiende la caminata un cierto número de pasos al este y un cierto número de pasos al norte |

El kickstart es auto explicativo, se hace hincapié en que el alumno sepa leer correctamente código.

Se entrega junto con el kickstart un main con una CLI (Command Line Interface o interfaz de línea de comandos) interactiva que ayudará en la prueba del programa.

El **ejercicio número 1** consiste en implementar el tipo abstracto de datos.

El **ejercicio número 2** consiste en completar apropiadamente el switch de la función `main` en los casos `case EMPTY` y `case QUIT`. En este punto deben prestar especial atención a los memory leaks sobre todo en este punto!!!

## Consideraciones:

- Se provee el archivo **`Makefile`** para facilitar la compilación.
- Se recomienda usar las herramientas **`valgrind`** y **`gdb`**. ☣️
- Se ofrece una carpeta **`outputs`** que tiene las salidas esperadas para ***make test*** y los ***valgrind*** 🛟
- Es muy importante y se evalúa comprensión y lectura de código.
- Entender las estructuras y tipos de datos dados es clave! Una vez analizado esto, el parcial es fácil, simplemente completar las funciones que aún no están implementadas.
- Si el programa no compila, no se aprueba el parcial.
- Los *memory leaks* bajan puntos.
- Entregar código muy improlijo puede restar puntos
- **No modificar los .h**☣️