
 <b>Universidad Nacional de La Matanza</b> 	Tópicos de programación 03635 Parcial Turno Tarde 24/06/2025
Apellido y Nombre:	
DNI:	
Calificación:	

## Parcial Tópicos de programación

Comisión: 02-2600

### Instrucciones de entrega

ATENCIÓN: para que el examen pueda ser corregido y archivado debe ser entregado obligatoriamente antes de la hora límite con el siguiente formato: DNI\_APELLIDO\_NOMBRE.zip, ejemplo 41127133\_PEREZ\_MARIA\_PIA\_P.zip. Observe que es un .zip, no es .rar ni .7z respete la forma de entrega. La P como sufijo indica que es el PARCIAL y la R como sufijo indica que es recuperatorio. Adjunte el enunciado y **elimine las carpetas bin y obj**, está compartiendo programas y por conocidas razones los servidores eliminarán comprimidos que contengan binarios o ejecutables.

## Enunciado

Se entrega junto con este documento un proyecto en `code::blocks`. El proyecto contiene una función genérica que puede encontrar un elemento en un archivo binario mediante una búsqueda binaria.

La función (Ver 1) recibe los siguientes argumentos:

- **clave:** Un puntero al elemento a buscar. Este elemento debe tener el o los campos clave asignados para la búsqueda. Es también el campo de ordenamiento del archivo.
- **nomarch:** El nombre del archivo donde se realizará la búsqueda.
- **tam:** El tamaño del elemento (Es una implementación genérica).
- **cmp:** Una función de comparación asociada al tipo y orden de los elementos.

En caso de encontrar el elemento los datos del mismo son copiados al buffer. Puede probarla para verificar su funcionamiento, se agrega un pequeño ejemplo de uso. Observe que el valor de retorno de la función indica si el elemento fue encontrado o si por el contrario no está en el archivo o la función no fue exitosa por algún otro motivo. Como sabemos, al ser búsqueda binaria, solo aplica a archivos ordenados por la clave de búsqueda.

Descomente la macro `#define EXAMEN` e implemente su propia versión de la función

de búsqueda binaria sobre archivos. Se pide una **búsqueda binaria**, no una búsqueda secuencial.

Listing 1: Prototipo funcion de búsqueda binaria en archivo.

```
int fbbinaria(void *clave ,
              const char* nomarch ,
              size_t tam ,
              int cmp(const void *, const void *))
```

Recuerde que su implementación debe funcionar para el ejemplo que se entrega o para cualquier otro ejemplo o tipo de dato. Es su responsabilidad probar que lo entregado cumple los requisitos. No modifique el prototipo. No se indica tamaño del archivo, el mismo puede ser lo suficientemente grande para no soportar ser cargado en memoria.

## Requerimientos funcionales

El resultado debe ser correcto. Si hubiese casos de prueba debe ser correcto para los casos entregados u otros casos posibles.

## Requerimientos no funcionales

El resultado es condición necesaria pero no suficiente. Se deben cumplir ademas los requerimientos no funcionales.

- Si se entrega un proyecto y se lo indica, no modificar la firma de las funciones.
- Vectores y cadenas de texto deberán ser manipulados utilizando aritmética de punteros.
- No se permite el uso de VLA (Variable length arrays)
- Declaraciones siempre al inicio de bloque por compatibilidad ANSI C.
- El sistema debe ser eficiente:
  - En el uso de memoria
    - No declare vectores o matrices auxiliares si no es necesario.
    - No cargue en memoria mas de lo necesario.
  - En cantidad de ciclos de procesador y en el caso de matrices las soluciones deben ser óptimas.
- No almacene archivos completos en memoria excepto que la problemática planteada lo exija (Normalmente la descripción del problema define que el conjunto de datos es conocido y finito).
- Utilice nombres descriptivos, separe en funciones y cuide la prolijidad.