1. Hacer un programa que defina arreglos nativos de 2 componentes de todos los tipos de variables que conoce (**`char`**, **`int`**, **`float`**, **`double`**) y de algunos UDT (**`Complex`**, **`Punto2D`**) e imprima la dirección de memoria de cada uno de los componentes de cada uno de los vectores. ¿Cuál es la distancia en bytes entre las componentes de cada vector? Comparar con lo que retorna el operador **`sizeof()`**.

2. Defina un UDT que no contenga atributos, solo funciones miembros. Instancie el tipo y compruebe que, si bien no tiene atributos, tiene memoria asociada. ¿Cómo lo justifica?

3. Implementar la función **`TransponeNN`** que reciba una matriz cuadrada como puntero al primer elemento de la matriz y la trasponga sobre sí misma. Utilizar el prototipo:

   ```
   void TransponeNN(double *matriz, int n);
   ```

   Ejemplo de uso:

   ```
   double m[10][10];
   ...
   TransponeNN(&m[0][0], 10);
   ```

4. Implementar la función **`CompareStrings`** que compare strings nativos *asciibéticamente,* devolviendo 0 si los strings son iguales, un valor positivo si el primero es mayor y un valor negativo si el segundo es mayor. Utilizar el prototipo:

   ```
   int CompareStrings(const char *str1, const char *str2);
   ```

5. ¿Por qué el siguiente código no es saludable? Describa el problema.

   ```
   int *getPtr() {
       int a = 4;
       return &a;
   }
   ```

6. ¿Por qué el siguiente código no es saludable? Describa el problema.

   ```
   void fun() {
       int a = 4;
       int *pa;

       *pa = a;
   }
   ```

7. ¿Qué hace el siguiente código? Compruebe su diagnóstico.

```
void fun() {
    int a[3][3] = { 0 };
    int *pFilas[3] = { &a[0][0], &a[1][0], &a[2][0] };
    pFilas[0][0] = pFilas[1][0] = pFilas[2][0] = 1;
}
```

8. Implemente la función **integra** que calcule el área bajo la curva definida por la función **fun** utilizando el método de trapecios. Este método consiste en dividir el intervalo *[a,b]* en subintervalos y aproximar la función en cada uno de ellos por una recta. Defina un criterio iterativo para ir cambiando la cantidad de subintervalos hasta lograr la convergencia. Prototipos:

```
using fun_ptr_t = double (*)(double);
double integra(double a, double b, fun_ptr_t fun);
```

https://es.wikipedia.org/wiki/Regla_del_trapecio

9. Escribir un programa que ordene un arreglo nativo de estructuras **Terna** por el campo **value** utilizando la función **qsort** e implementando una función de comparación adecuada.

```
struct Terna {
    int i;
    int j;
    int value;
};
```

Prototipo de **qsort (busque la documentación de referencia en la WEB)**:

```
#include <cstdlib>

qsort(void *base, size_t nmemb, size_t size,
      int (*compar)(const void *, const void *) );
```

¿Qué función de comparación usaría si quiere que el ordenamiento sea por la distancia entre (**i**,**j**) y el origen?

10. Escriba un programa que defina un arreglo nativo de 5 punteros a función del tipo:

```
typedef void (*FunPtr_t)(void);
```

Instancie el arreglo para que apunte a 5 funciones distintas que lo que hagan es anunciarse con su nombre. Active las funciones dentro de un loop generando un número al azar **X** y activando la función que está en la posición **X % 5** del vector de funciones.