Práctico 0

- TIPOS DE DATOS -

NOTA: Debe entregar el código completo del práctico en un único proyecto C++ (siempre debe hacer un *clean* del *workspace* antes de comprimir su contenido para entregar), el nombre del proyecto debe ser **edya**. Recuerden seguir los criterios aconsejados por la cátedra y respetar los prototipos de las funciones.

EJERCICIO 1

Para su resolución:

- Es necesario hacer uso de la función `malloc`.
- No deben utilizarse las librería provista por el lenguaje <string.h>, <cstring>, <stdlib.h> y <cstdlib>.

El ejercicio requiere implementar funciones que permitan realizar las siguientes operaciones sobre cadenas de texto (C String):

edya_strlen

```
/**
 * @brief Retorna el largo de una cadena
 * edya_strlen(NULL) = 0
 * edya_strlen("") = 0
 * edya_strlen("12345") = 5
 * @param str Cadena
 * @return size_t
 */
size_t edya_strlen(char * str);
```

edya_strcat

```
/**
 * @brief Concatena la cadena str1 y str2 en una nueva cadena.
 * edya_strcat("abc","xyz")  > "abcxyz"
 * edya_strcat(NULL, "abc")  > "abc"
 * edya_strcat("abc", NULL)  > "abc"
 * edya_strcat("", "")       > ""
 * edya_strcat(NULL, NULL)   > ""
 * @param str1
 * @param str2
 * @return char * (Nueva cadena)
 */
char * edya_strcat(char * str1, char * str2);
```

edya_strcpy

```
/**
 * @brief Copia la cadena apuntada por source en la cadena apuntada por
 * destination. Antes de hacer la llamada el puntero destination debe contener
 * suficiente espacio para copiar sobre la zona de memoria a la que apunta,
 * de lo contrario el comportamiento no queda definido.
 * @param value
 * @param str
 * @param base
 * @return char* (destination)
 */
char * edya_strcpy(char * destination, char * source);
```

edya_itoa

```
/**
 * @brief Convierte un entero positivo en una cadena numérica
 * según la base.
 * @param value
 * @param str
 * @param base Posibles valores [2,8,10,16]
 * @return char*   (str)
 */
char * edya_itoa(unsigned int value, char * str, int base);
```

EJERCICIO 2

Se requiere implementar un nuevo tipo de dato para representar una fecha según el calendario gregoriano. La estructura de datos debe llamarse **Date** y sus campos son **day**, **month** y **year**. Codifique funciones asociadas a este nuevo tipo de dato.

leapYear

```
/**
 * @brief La función calcula si un año es bisiesto
 * @param y Date
 * @return bool
 */
bool leapYear(unsigned int y);
```

leapYear

```
/**
 * @brief La función calcula si una fecha pertenece a un año bisiesto
 * @param d Date
 * @return bool
```

```
 */
bool leapYear(Date d);
```

datecmp

```
/**
 * @brief Retorna un valor entero indicando la relación entre las fechas:
 *  <0  la fecha d1 es menor que la fecha d2
 *  0   la fecha d1 es igual que la fecha d2
 *  >0  la fecha d1 es mayor que la fecha d2
 * @param d1
 * @param d2
 * @return int
 */
int datecmp(Date d1, Date d2);
```

dayOfYear

```
/**
 * @brief Retorna el día del año, un valor entre 1 y 365 (366 para
 * años bisiestos)
 * @param d
 * @return unsigned int
 */
unsigned int dayOfYear(Date d);
```

EJERCICIO 3

En una función `test_pointer()` declare las siguientes variables:

1. Un char.
2. Un puntero a un char.
3. Un array de 9 punteros a int.
4. Un puntero a un array de 9 int.
5. Un puntero a un puntero int.
6. Una matriz de 4 filas y 3 columnas
7. Un puntero a una matriz de int de 4 filas y 3 columnas.
8. Un array de N punteros a Date.

Usar depuración, volcado de memoria, asignaciones y sizeof para verificar la correcta definición.

EJERCICIO 4

Redefina la implementación de la función strlen del **ejercicio 1** utilizando aritmética de punteros (no puede usar tipos de datos enteros como char, int, short o long para recorrer la cadena).

```
/**
 * @brief Retorna el largo de una cadena
 * strlen(NULL) = 0
```

```
 * strlen("") = 0
 * strlen("12345") = 5
 * @param str Cadena
 * @return size_t
 */
size_t edya_ptr_strlen(char * str);
```

EJERCICIO 5

Partiendo del siguiente código liste todos los números de la matriz fila por fila utilizando **únicamente un for**.

```
int ROWS = 3, COLS = 4;
int num[ROWS][COLS] = {
        {1, 2,  3,  4},
        {5, 6,  7,  8},
        {9, 10, 11, 12}
    };
// pointer
int *ptr = &num[0][0];
// print
for(/**/){
    /**/
}
```

Salida esperada:
1 2 3 4 5 . . . 11 12