Práctico 1

- TIPOS DE DATOS -

NOTA: Debe entregar el código completo de las librerías (`date.h`, `word.h` y `edya_array.h`) y para el ejercicio 5 un documento de texto en formato Markdown. Recuerden seguir los criterios aconsejados por la cátedra y respetar los prototipos de las funciones. **Colocar en el encabezado de todos los archivos el nombre del Grupo y los miembros**.

EJERCICIO 1

Defina dentro de la librería **`date.h`** un tipo de dato enumerado **Month** que contenga identificadores para todos los meses del año (`1 JANUARY, 2 FEBRUARY, 3 MARCH, 4 APRIL, 5 MAY, 6 JUNE, 7 JULY, 8 AUGUST, 9 SEPTEMBER, 10 OCTOBER, 11 NOVEMBER, 12 DECEMBER`). Incluya también funciones que ayuden a la manipulación de este tipo de dato:

1. retornar el mes siguiente:
```
Month next(Month m)
```
2. retornar el mes anterior:
```
Month previous(Month m)
```
3. retornar el número del mes [1-12]:
```
unsigned short to_int(Month m)
```
4. retornar una cadena con el nombre del mes:
```
char* to_str(Month m)
```
5. retornar el mes a partir del valor numérico del mes:
```
Month to_month(unsigned short m)
```

EJERCICIO 2

Defina dentro de la librería **`date.h`** una estructura de datos (**Date**) que permita almacenar una fecha con los campos **day**, **month** y **year**. Los valores por defecto para la inicialización de la estructura son: {1, JANUARY, 1970}
Codifique allí también las funciones asociadas a este nuevo tipo de dato:

1. convertir una fecha en una cadena (con el formato dd/mm/yyyy):

```
/**
 * @brief Convierte un Date en una cadena con el formato dd/mm/yyyy
 * @param d
 * @return char *
 */
char* to_str(Date d);
```

2. Cargar una fecha a partir de una cadena de caracteres de formato dd/mm/yyyy[1]

```
/**
 * @brief Devuelve una fecha a partir de una cadena de caracteres
 * de formato dd/mm/yyyy
 * @return Date
 */
Date to_date(char *str);
```

3. Verificar si es año bisiesto:

```
/**
 * @brief Retorna si es año bisiesto
 * @param d
 */
bool leapYear(Date d);
```

4. Verificar que es una fecha correcta:

```
/**
 * @brief Retorna true si una fecha es correcta
 * @return bool
 */
bool isValidDate(Date date)
```

5. comparar dos fechas retornando la cantidad de días entre ellas.

```
/**
 * @brief Retorna la cantidad de días entre la fecha dt1 y dt2.
 * Si dt1 es mayor a dt2 retorna un valor negativo
 * @param d
 * @return int
 */
int getDifference(Date dt1, Date dt2);
```

EJERCICIO 3

Una palabra en una computadora es una cadena finita de bits (0|1) que son manejados en forma conjunta. Defina dentro de la librería **word.h** una estructura de datos **word** de 32 bits (4 bytes).

- Estudie los tipos de datos primitivos[2] y sus modificadores[3] para encontrar el más adecuado para la implementación.
- Utilizando uniones y estructuras defina el tipo de dato **word** de forma que permita hacer operaciones sobre toda la palabra o sobre un byte en particular.

[1] Se respeta el formato, en caso de que la fecha no sea correcta poner por defecto 01/01/1970.
[2] https://en.cppreference.com/w/cpp/language/types
[3] https://en.cppreference.com/w/cpp/language/types#modifiers

Código de ejemplo para el uso del TDA word:

```
word w = {0xA0B1C2D3};
cout << endl << "word: "<< hex << w.data;
w.b0 = 0xD3;
w.b1 = 0xC2;
w.b2 = 0xB1;
w.b3 = 0xA1;
cout << endl << "b0: "<< hex << int(w.b0);
cout << endl << "b1: "<< hex << int(w.b1);
cout << endl << "b2: "<< hex << int(w.b2);
cout << endl << "b3: "<< hex << int(w.b3);
```

Salida por consola:

```
word: a0b1c2d3
b0: d3
b1: c2
b2: b1
b3: a1
```

Analizar el contenido de la memoria ram donde se aloja la variable **w** del tipo **word** y explique qué relación existe entre las direcciones de memoria de los campos data, b0, b1, b2 y b3.

EJERCICIO 4

Utilizando el tipo de dato **Date** desarrolle una función que ordene un arreglo de tamaño N en forma ascendente o descendente.

- Use reserva de memoria dinámica para definir a partir de N el tamaño del arreglo en tiempo de ejecución.
- Comentar: ¿Cómo se pueden hacer distintos ordenamientos sin modificar el arreglo original y sin crear una copia completa de la estructura (“no programar”)?

```
/**
 * @brief Ordena en forma ascendente ("asc") o
 * descendente ("desc") un arreglo de N estructuras del tipo Date
 * @param dates
 * @param N
 * @param o
 */
void sort(Date *dates, size_t N, orden o=asc);
```

EJERCICIO 5

Desarrolle brevemente qué son los campos de bits, identifiqué aspectos positivos y negativos de su uso y para finalizar desarrolle un ejemplo de libre elección donde se pueda analizar su uso.

EJERCICIO 6

Defina dentro de la librería **edya_array.h** una estructura de datos (**EArray**) que permita almacenar un array de enteros de tamaño dinámico. Los atributos de la estructura son **array** y **size**.

Codifique allí también las funciones asociadas a este nuevo tipo de dato:

1. Crear un array de tamaño N.

```
EArray createArray(size_t n);
```

2. Redimensionar el array

```
void resize(EArray &earray, size_t new_n);
```

3. ¿Qué otra función considera apropiada para implementar? Programar la función sugerida.