# TP N°2 – Arrays

## Parte Práctica

**Primer parte: Arrays unidimensionales.**

**Ejercicio 1**: Desarrolle una función que pida al usuario valores de temperatura en Kelvin (T>=0) y las almacene en un array. El ingreso de datos finaliza cuando se ingresa un valor negativo ó cuando se llega al máximo de valores. El prototipo es el siguiente:

```
void ingreso_temps(float temps[], int len);
```

Donde:

- `temps[]`: array donde se guardarán las temperaturas ingresadas.
- `len`: largo máximo del array.

Nota:

El valor posterior al último ingresado será un número negativo.

**Ejercicio 2**: Desarrolle una función que imprima un array de temperaturas, sabiendo que el mismo termina en un número negativo. El prototipo es el siguiente:

```
void print_temps(float temps[]);
```

Donde:

- `temps[]`: array de temperaturas.

**Ejercicio 3**: Desarrolle una función que calcule y devuelva el promedio de las temperaturas incluidas en un array, tomando como válidos solo los valores entre un mínimo y un máximo. Los valores fuera del rango deben copiarse a otro array. El prototipo es el siguiente:

```
float calcula_promedio (float temps [] , int min, int max, float rangeout[]);
```

Donde:

- `temps[]`: array con las temperaturas en Kelvin (T>=0).
- `min`: valor mínimo a incluir en el promedio.
- `max`: valor máximo a incluir en el promedio.
- `rangeout[]`: array donde se copiaran las temperaturas fuera de rango.

Devuelve:

un valor negativo si el array empieza con un número negativo (array vacío), si todos los valores están fuera del rango ó si `min` > `max`.
Caso contrario, el promedio.

Nota:

Se desconoce el largo del array temps, pero se sabe que termina con un valor negativo.
`rangeout` debe terminar en un valor negativo.
Si `min` ó `max` son negativos, se tomarán para el promedio **todas** las temperaturas del array.
El largo del array `rangeout` es igual al de temps.

**Ejercicio 4**: Desarrolle un programa que, utilizando las funciones creadas anteriormente, pida al usuario una serie de temperaturas en Kelvin, extraiga las que estén fuera del rango [MIN,MAX], calcule el promedio de los valores válidos y luego imprima el array original, el promedio calculado y el array con los valores fuera de rango.

Nota:

LEN (largo de los arrays), MIN y MAX serán constantes simbólicas.
Si el array original comienza con un número negativo se imprime “No hay datos”.
Si todos los valores están fuera de rango se imprime “Todos los datos son inválidos”.

**Segunda parte: Arrays bidimensionales – Matrices.**

Ejercicio 1: Desarrolle una función que reciba como parámetro un array de datos, verifique si con el puede formarse una matriz de NxN (cuadrada) y en caso afirmativo realice la copia de los datos. El prototipo es el siguiente:

```
int crear_matriz(float datos[], float matriz[][N]);
```

Donde:

`datos`: es un array floats >= 0 que termina en un número negativo.
`matriz`: es donde deben copiarse los datos en caso que corresponda.
`N`: es una constante simbólica que representa la tamaño de la matriz.

Devuelve:

-1 si el array está vacío.
-2 si no puede formar una matriz de NxN.
0 si todo sale bien.

Nota:

Recuerde que no debe comparar dos variables de tipo float por igualdad.

**Ejercicio 2**: Desarrolle una función que trasponga una matriz cuadrada pasada como parámetro. El prototipo es el siguiente:

```
void trasponer_matriz(float matriz[][N]);
```

Donde:

`matriz`: es la matriz a trasponer.
`N`: es una constante simbólica que representa la dimensión de la matriz.

**Ejercicio 3**: Desarrolle una función que imprima una matriz cuadrada. El prototipo es el siguiente:

```
void print_matriz(float matriz[][N]);
```

Donde:

`matriz`: es la matriz a imprimir.
`N`: es una constante simbólica que representa la dimensión de la matriz.

**Ejercicio 4**: Desarrolle un programa que, utilizando las funciones creadas anteriormente, pida al usuario una serie de valores finalizados en un número negativo, verifique si puede formarse una matriz de NxN y en caso de ser posible, trasponga dicha matriz. Luego imprima la matriz original y la traspuesta.

Nota:

Para el ingreso de datos, puede utilizar la función creada en la sección 1.

**Notas generales**:

Si lo necesita puede crear funciones auxiliares para simplificar las funciones más generales, por ejemplo para obtener el largo de un array terminado en número negativo.
Debe separar el código en los siguientes archivos:

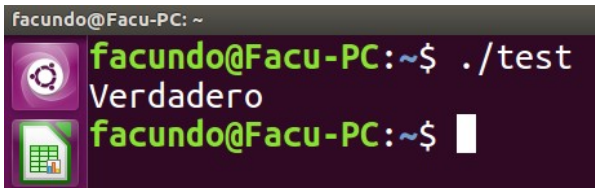
1. funciones_tp2.c : Contiene todas las funciones pedidas.
2. funciones_tp2.h : Contiene todos los prototipos, constantes simbólicas, etc.
3. main_arrays.c : Contiene el main de la sección 1.
4. main_matrices.c : Contiene el main de la sección 2.
5. compilar: Script para automatizar la compilación.

Se entregarán solamente los archivos mencionados en el ítem anterior dentro de una carpeta con el siguiente nombre: tp2_apellido. **NO** incluir archivos objeto ni ejecutables (excepto compilar).

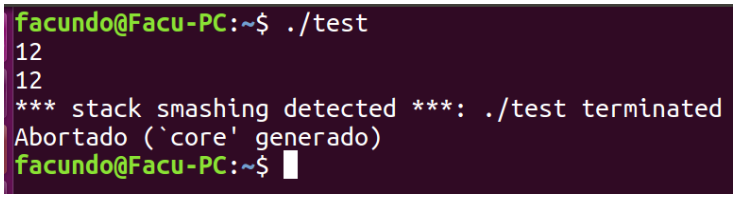
## Parte Teórica

Explique por qué los siguientes bloques de código generan esas salidas.

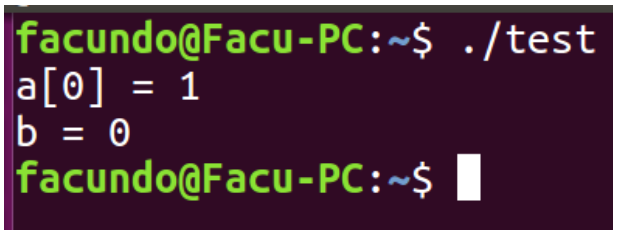
```c
#include <stdio.h>

int main (void){
    char a=255;

    if(a==-1)
        puts("Verdadero");
    else
        puts("Falso");

    return 0;
}
```




```c
#include <stdio.h>

int main (void){
    int a[10] = {0};

    scanf("%d", &a[10]);
    printf("%d\n", a[10]);

    return 0;
}
```




```c
#include <stdio.h>

void incrementa_array(int a[]){
    a[0]=a[0]+1;
}
void incrementa_int(int b){
    b++;
}


int main (void){
    int a[1] = {0};
    int b = 0;

    incrementa_array(a);
    incrementa_int(b);

    printf("a[0] = %d\n", a[0]);
    printf("b = %d\n", b);

    return 0;
}
```




¿Por qué se incrementa el valor dentro del array pero no 'b'?