

Nombre:

Apellidos:

Grupo: ☐ GITI 1E ☐ GITI 1F ☐ GITI-1G ☐ GITI-1H

## CONTESTAR EL EXAMEN EN EL LIBRILLO DE RESPUESTAS

### Normas para el examen de Fundamentos de Informática:

- 1) **TODOS** los cuadernillos de respuestas y enunciados, deberán ser entregados y estar debidamente cumplimentados con el nombre y apellidos del alumno, así como al **grupo** al que pertenece. **(Cualquier cuadernillo de respuesta no entregado o sin cumplimentar, será calificado con un 0)**
- 2) Solo se permite tener sobre la mesa **el enunciado** y **uno de los cuadernillos de respuesta**. (el resto de cuadernillos de respuesta deberán estar ocultos bajo el enunciado o el cuadernillo de respuesta sobre el que se está contestando, si no se le podrán retirar al alumno).
- 3) **No se pueden desgrapar** ni el enunciado ni el cuadernillo de respuestas.
- 4) **Recordatorio:** Copiar en un examen puede llegar a implicar la pérdida automática de las convocatorias Ordinaria y Extraordinaria.
- 5) “No está permitida la consulta de libros o apuntes, ni el uso de calculadoras programables”
- 6) **La cajonera** de la mesa **deberá permanecer vacía**.
- 7) Todo **dispositivo electrónico** con capacidad de comunicación (teléfonos móviles, relojes inteligentes, dispositivos Bluetooth, etc.) deberá permanecer **apagado y fuera del alcance del alumno** (en el interior de una bolsa o mochila cerrada que se depositará en un extremo del aula o debajo de la silla del alumno según indique el Profesor) durante la realización del examen.

**Duración: 1 h y 30 minutos**

**Programa 1: Polinomio de Chebyshev (3 puntos)**

Escribir un **programa principal** que calcule el valor del polinomio de Chebyshev de grado  $n$  para todos los valores de  $x$  en el intervalo  $[-1, 1]$ , en pasos de  $x$  de 0.1 (-1; -0.9; -0.8; ...; 0.8; 0.9; 1).

Para ello, el programa pedirá al usuario el grado  $n$  del polinomio que desea calcular (**NO es necesario validar**).

A continuación, calculará de forma iterativa y mostrará por pantalla el valor de dicho polinomio para cada valor de  $x$  entre -1 y 1, a paso de  $x$  de 0.1 (-1; -0.9; -0.8; ...; 0.8; 0.9; 1).

Cálculo de los Polinomios de Chebyshev:

Para  $x$ :  $-1 \leq x \leq 1$

Los primeros polinomios son:

$$T_0(x) = 1$$

$$T_1(x) = x$$

El resto de polinomios se calculan mediante la expresión:

$$T_n(x) = 2 \cdot x \cdot T_{n-1}(x) - T_{n-2}(x)$$

Ejemplos:

$$T_2(x) = 2x^2 - 1$$

$$T_3(x) = 2 \cdot x \cdot (2x^2 - 1) - x = 4x^3 - 3x$$

$$T_4(x) = 2 \cdot x \cdot (4x^3 - 3x) - (2x^2 - 1) = 8x^4 - 8x^2 + 1$$

$$T_5(x) = 2 \cdot x \cdot (8x^4 - 8x^2 + 1) - (4x^3 - 3x) = 16x^5 - 20x^3 + 5x$$

$$T_6(x) = \dots$$

## Programa 2: Carrera de primavera (3 puntos)

En el circuito universitario de campo a través de la Universidad Pontificia Comillas se realiza una prueba de primavera cada curso de 10 km de distancia.

Cada corredor obtiene una puntuación en función de los minutos y segundos empleados en finalizar la prueba (se supone que ningún corredor tarda más de una hora en finalizar la prueba). Así, si un corredor ha tardado **ab** minutos y **cd** segundos en finalizar la carrera, la puntuación de este corredor es:

$$\text{puntos} = \frac{1000}{abcd}$$

Por ejemplo, si un corredor ha empleado un tiempo de 29 minutos y 28 segundos, su puntuación será:

$$\text{puntos} = \frac{1000}{2928} = 0.34$$

Como realizar este cálculo a mano para todos los participantes es una tarea tediosa y propensa a errores, el servicio de deportes de la Universidad le ha encargado que realice un programa para facilitar esta tarea. El programa debe ir pidiendo sucesivamente los minutos y segundos empleados por cada corredor (guardándolos en dos variables diferentes que **NO es necesario validar**), calcular sus puntos obtenidos y debe guardar la puntuación de los tres mejores corredores, ordenados de mayor a menor.

Dicha puntuación se calculará mediante la siguiente función (que se debe implementar):

**float CalPuntos (int minutos, int segundos);**

La lectura de minutos y segundos de todos los corredores finaliza cuando se introduzca el valor -1 en la lectura de los minutos de un corredor.

El programa finalizará mostrando por pantalla la puntuación obtenida por los tres ganadores de la prueba, por ejemplo:

Puntuación primero: 0.34 puntos

Puntuación segundo: 0.28 puntos

Puntuación tercero: 0.24 puntos

**Nota:** NO se pueden usar vectores para guardar las puntuaciones y los minutos y segundos de TODOS los corredores.

### Programa 3: Rotación de un vector (4 puntos)

Se desea desarrollar un programa en lenguaje C en el que dado un vector de valores enteros de tamaño máximo 50, se pida al usuario si desea rotarlo a la izquierda o a la derecha, así como el número de posiciones a rotar respecto a la composición del vector original.

Se define como rotación de un vector el desplazamiento de una posición de todos sus elementos a la izquierda o a la derecha. Por ejemplo, si se dispone del vector  $V = (3, 7, 4, 8, 9, 10)$  y se desea rotarlo a la derecha 2 posiciones, quedaría:

	Rotación derecha						
Rotar derecha							
Vector original	3	7	4	8	9	10	
1ª Rotación derecha	10	3	7	4	8	9	
2ª Rotación derecha	9	10	3	7	4	8	

Se pide desarrollar un programa principal **main()** que solicite al usuario un vector de números enteros mediante la llamada a la función **Rellenar()** que permite guardar una serie de valores enteros en un vector y devuelve el número de valores introducidos por el usuario. El prototipo de esta función es:

```
int Rellenar (int vector []);
```

Esta función **NO HAY que implementarla** sino solo llamarla desde el programa principal **main()**.

Una vez obtenido el vector con sus valores, se solicitará al usuario desde el programa principal que introduzca si desea rotar el vector a la izquierda ('I') o a la derecha ('D'), así como el número de posiciones a rotar (positivo, no puede ser mayor que el número de elementos del vector). **Sí hay que validar ambas variables.**

A continuación, se deberá llamar a una función que se encargue de rotar el vector una sola posición y esta operación se realizará cuantas veces sea necesaria. No se permitirá el uso de ningún otro vector auxiliar, solo se debe usar un único vector para lograr el objetivo buscado.

El prototipo de la función **Rotar()** es el siguiente:

```
void Rotar(int vector [], char direccion, int dimension);
```

Aquí **vector** es el vector a rotar, **direccion** si se ha de hacer esa rotación a la derecha o a la izquierda y **dimension** es el tamaño del vector.

Cada vez que se llame a la función **Rotar** se presentará después cómo ha quedado el vector mediante la función **Mostrar** cuyo prototipo es el siguiente:

```
void Mostrar(int vector[], int dimension);
```

Aquí **vector** y **dimension** tienen el mismo significado que en la función **Rotar()**.

Aparte del programa principal **main** se han de desarrollar también las funciones **Rotar()** y **Mostrar()**.

	<h1>Fundamentos de Informática</h1> <h2>1º GITI</h2>	<h3>Mar / 2022</h3>
--	--	---------------------

Sigue un ejemplo de cómo sería el resultado que un usuario podría ver en un caso ejemplo:

Desea rotar a la izquierda(I) o a la derecha(D): I

Cuántas posiciones desea rotar: 3

Vector original

=====

3 7 4 8 9 10 23 45 67 89

Vector rotado 1 posición

=====

7 4 8 9 10 23 45 67 89 3

Vector rotado 2 posición

=====

4 8 9 10 23 45 67 89 3 7

Vector rotado 3 posición

=====

8 9 10 23 45 67 89 3 7 4

# Ayuda para la programación en C

## Estructura de un programa C

```
/*
Programa de Ejemplo
Fecha_
Autor_
*/
#include _____
#define _____
typedef _____
[Prototipos]

void main(void)
{
    [variables] /* descripción */

    [instrucciones]
}
```

## Caracteres especiales

'\n' cambio de línea (newline)  
'\r' retorno de carro  
'\0' carácter 0 (NULL)  
'\t' TAB  
'\'' comilla simple '  
'\"' comilla doble "  
'\\' la barra \

## Formatos de printf y scanf

%d int  
%hd short  
%ld long  
%u unsigned int  
%hu unsigned short  
%lu unsigned long  
%f float, double  
%lf double (sólo scanf)  
%c char  
%S cadena de caracteres

## Operadores

Aritméticos int: + - \* / %  
Aritméticos double: + - \* /  
Otros aritméticos: ++ -- += -= \*= /=  
Lógicos y relacionales:  
> < >= <= == != && || !

## Bucles

### Bucle for

```
for (inicialización, condición, instrucción_final)
{
    [instrucciones]
}
```

Ejemplo: for(i=0; i<10; i++)

### Bucle while

```
while (condición) {
    [instrucciones]
}
```

### Bucle do-while

```
do {
    [instrucciones]
} while(condición);
```

## Bloque if

### caso 1:

```
if (condición) {
    [instrucciones]
}
```

### caso 2:

```
if (condición)
{
    [instrucciones_1]
} else {
    [instrucciones_2]
}
```

### caso 3:

```
if (condición_1) {
    [instrucciones_1]
} else if (condición_2) {
    [instrucciones_2]
    ...
} else if (condición_n)
{
    [instrucciones_n]
} else {
    [instrucciones]
}
```

## Sintaxis del switch

```
switch(expresión_entera) {
case constante_1:
    [instrucciones_1]
    break;
case constante_2:
    [instrucciones_2]
    break;
...
case constante_3:
    [instrucciones_3]
    break;
default:
    [instrucciones]
}
```

## Vectores y matrices

```
double vector[10];
char cadena[256];
char matriz[10][20];
```

```
vector[2]=3;
scanf("%lf", &vector[7]);
```

## Cadenas de caracteres

```
char cadena[N];
```

### Lectura:

```
scanf("%s", cadena);
    lee una palabra
```

```
gets(cadena);
    lee una frase hasta fin de línea
```

```
fgets(cadena, N, stdin);
    lee una frase con control de tamaño. También lee '\n'
    si le cabe
```

### Escritura:

```
printf("%s", cadena);
    escribe una cadena por pantalla, vale para frase o
    palabra
```

## Funciones estándar de string.h

```
size_t strlen( char *str );
    devuelve la longitud de la cadena
```

```
strcpy(char *to, char *from);
    copia o inicializa
```

```
int strcmp(char *s1, char *s2);
    compara las cadenas s1 y s2
    0 → s1 es igual a s2
    <0 → s1 es menor que
    s2 → 0 s1 es mayor
    que s2
```

# Ayuda para la programación en C

## Funciones

### Prototipo:

*tipo* NombreFun(*tipo* var1, ... , *tipo* varN);

### Estructura de la función:

*tipo* NombreFun(*tipo* var1, ... , *tipo* varN)

/\* Descripción general

Argumentos: ...

Valor Retornado: ...

Advertencias de uso: ...

\*/

{

    [variables locales]

    [instrucciones]

    return *expresión*;

}

### Ejemplos de prototipos y llamadas:

int Sumar(int a, int b);

void Cambio(int \*a, int \*b);

double CalcularMediana(double a[], int

n); float Traza(float mat[][20], int n,

int m);

res=Sumar(x,y);

Cambio(&x, &y);

med=CalcularMediana(vec,n);

tra=Traza(mat,n,m);

## BORRADOR

(Esta hoja no será corregida, NO SE  
PUEDE DESGRAPAR

# **BORRADOR**

(Esta hoja no será corregida, NO SE PUEDE DESGRAPAR)