

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). Los cuadernillos de respuesta **no se podrán desgrapar**.
- 3) **Recordatorio**: Copiar en un examen puede llegar a implicar la pérdida automática de las convocatorias Ordinaria y Extraordinaria.
- 4) "No está permitida la consulta de libros o apuntes, ni el uso de calculadoras programables"
- 5) **La cajonera** de la mesa **deberá permanecer vacía**.
- 6) 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.

Programa 1: Cálculo del seno en un intervalo de ángulos (3 puntos)

Realizar un programa que pida un ángulo en grados y el número de puntos del eje x, y seguidamente llame a la función `Mostrar()` la cual pintará el seno en dicho intervalo. Cada vez que se llama a la función `Mostrar()` se pinta un punto donde `x` es el valor del ángulo, mientras que `y` es el seno de dicho ángulo.

Para ello, en el programa principal, se solicitará al usuario que introduzca un ángulo en grados, `x_grad` (no hace falta validar), el cual habrá que convertir a radianes (`x_rad`), sabiendo que 180 grados son π radianes. Se tiene que definir la constante simbólica `PI` con el valor 3.1415626.

A continuación, se solicitará al usuario el número de coordenadas `num` que tendrá el eje x y que serán equidistantes (validando que esté entre 0 y 100), que irá de 0 grados al ángulo en radianes. Cada posición del vector x (`vector_x`) será una coordenada, mientras que su correspondiente en el vector y (`vector_y`) será su seno. Ver ejemplos.

Una vez cargados los dos vectores `vector_x` y `vector_y`, se llamará sucesivamente a la función `Mostrar()`, que **NO HAY QUE CODIFICAR** cuyo prototipo es:

```
void Mostrar(float x, float y);
```

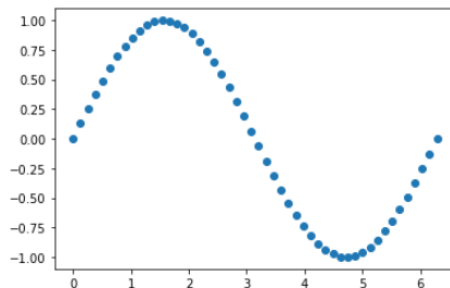
Esta función cada vez que es llamada pinta un punto en las coordenadas x,y

En este ejercicio hay que realizar el programa completo salvo la codificación de la función `Mostrar()`.

Ejemplo 1:

Introduzca un ángulo `>0`: 360

Introduzca el número de coordenadas del eje x: 50



Aclaración, NO SE PIDE MOSTRAR VECTORES

	x[0]				x[49]			
Contenido del vector x:	0.0	0.128	0.256	...	6.283			
	y[0]				y[49]			
Contenido del vector y:	0.0	0.127	0.253	...	0.000			

Programa 2: Función Mi_Substr() (3,5 puntos)

Se pide **codificar sólo una función** `Mi_Substr()` que lo que hace es obtener una subcadena de la cadena de caracteres original introducida por el usuario, según una serie de criterios. **No es necesario codificar el `main()`**. Veamos primeramente los parámetros que recibe esta función cuyo prototipo es:

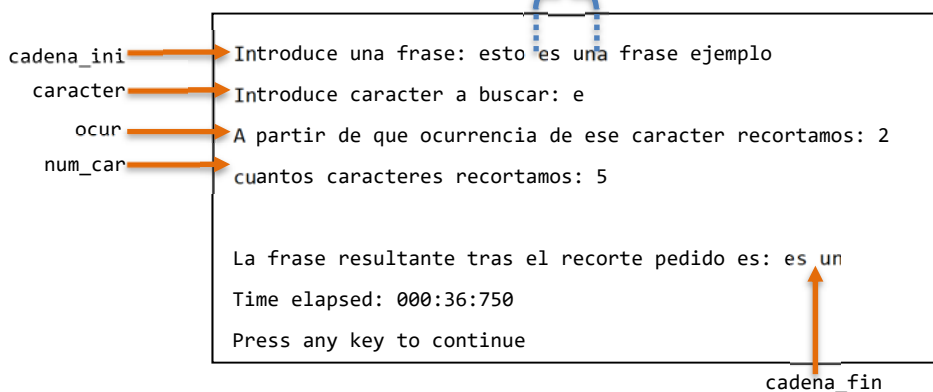
```
int Mi_Substr(char cadena_ini[], char caracter, int ocur, int num_car, char
cadena_fin[]);
```

- `cadena_ini` : un vector de caracteres estático de N elementos, siendo N una constante simbólica de valor 100, que representa la cadena inicial introducida por el usuario. **Por simplificación se asume que todos los caracteres de la frase introducida son minúsculas.**
- `caracter` : variable de tipo carácter que representa el carácter que vamos a buscar en la cadena inicial (`cadena_ini`), a partir del cual realizaremos y obtendremos la subcadena que queremos.
- `ocur` : variable de tipo entero que representa la ocurrencia del carácter anterior a partir del cual obtendremos la subcadena. Por ejemplo, si el carácter fuera una 'a' puede darse el caso de que haya más de una en la frase inicial y por tanto deberíamos indicar con esta variable a cuál de ellas nos referimos.
- `num_car` : variable de tipo entero que representa el número de caracteres que tomaremos de la cadena inicial para formar la subcadena, a partir de la ocurrencia del carácter proporcionado por el usuario.
- `cadena_fin` : vector de caracteres estático de N elementos que representa la subcadena que queríamos obtener.

La función devuelve el valor -3, si `caracter` (el carácter introducido) no se encuentra en la `cadena_ini`. La función devolverá -2, si hay menos ocurrencias del carácter introducido que la ocurrencia que nos solicitan en la variable `ocur`. Devolverá -1 si una vez se ha comprobado que existe la ocurrencia del carácter pedido, no hay suficientes caracteres (`num_car`) desde el encontrado hasta el final de la cadena. Por último, en caso de que todo vaya bien, la función devolverá 1 y la subcadena que va desde la ocurrencia pedida (`ocur`) hasta tener un total de `num_car` caracteres **incluyendo el carácter buscado**.

Ejemplos de ejecución:

Substring= a partir de la 2ª ocurrencia de la letra e tomamos 5 caracteres contando la e y el espacio en blanco



cadena_ini → Introduce una frase: esto es una frase ejemplo

caracter → Introduce caracter a buscar: e

ocur → A partir de que ocurrencia de ese caracter recortamos: 2

num_car → cuantos caracteres recortamos: 5

La frase resultante tras el recorte pedido es: es un

Time elapsed: 000:36:750

Press any key to continue

cadena_fin

```
Introduce una frase: Hola que tal estas
Introduce caracter a buscar: k
A partir de que ocurrencia de ese caracter recortamos: 3
cuantos caracteres recortamos: 2

El caracter k no está en la cadena
Time elapsed: 000:22:469
Press any key to continue
```

```
Introduce una frase: Hola que tal
Introduce caracter a buscar: a
A partir de que ocurrencia de ese caracter recortamos: 2
cuantos caracteres recortamos: 8

La frase termina antes de los caracteres pedidos
Time elapsed: 000:13:094
Press any key to continue
```

Programa 3: Selección aleatoria para control de calidad (3,5 puntos)

Para llevar a cabo el plan de control de calidad de una fábrica de motores, se precisa un programa en C que seleccione aleatoriamente una pieza de cada tipo de motor de los que se fabrican.

Para ello, se dispondrá de un vector que contendrá los códigos de todas las piezas de todos los motores que se fabrican. Los tipos de motor son números consecutivos, comenzando por 1, habiendo siempre piezas para cada tipo de motor. El tipo de motor al que pertenece una pieza se obtiene dividiendo el código de pieza entre 100. En la figura ejemplo, para un posible vector de códigos de piezas, el elemento del vector con índice 0 tiene como código de pieza 304, y el tipo de motor es 3 ($304/100=3$).

v_codigos	304	405	487	532	403	128	288	...	245	119	367	201	582	127
Índices del vector	0	1	2	3	4	5	6	...	114	115	116	117	118	119

Ejemplo de vector para 5 tipos de motores y 120 códigos de piezas

El programa irá seleccionando elementos del vector de forma aleatoria hasta obtener un código de pieza de cada tipo de motor. El primer elemento elegido al azar del vector intercambiará su valor con el primer elemento de este mismo vector. El segundo elemento elegido se intercambiará con el segundo elemento del vector y, así, sucesivamente. Deberá comprobarse, antes de intercambiar el valor de cada elemento elegido, que el tipo de motor del actual elemento elegido al azar no coincida con el tipo de motor de los elementos ya elegidos. En caso de ya existir, deberá seleccionarse otro elemento del vector al azar hasta que el tipo de motor no se encuentre entre los ya elegidos. Ejemplo:

- Suponga como vector inicial el que se muestra en la figura y que el primer elemento del mismo elegido al azar ha resultado el que tiene como índice 114. Este elemento deberá intercambiar su valor con el valor del primer elemento del vector. Al ser el primer elemento elegido, no haría falta comprobar que el tipo de motor (tipo 2 en este caso) ya ha sido elegido:

v_codigos	304	405	487	532	403	128	288	...	245	119	367	201	582	127
Índices del vector	0	1	2	3	4	5	6	...	114	115	116	117	118	119

- El vector quedaría actualizado de la siguiente forma, donde el elemento con índice 0 corresponde al primer código de pieza elegido al azar:

v_codigos	245	405	487	532	403	128	288	...	304	119	367	201	582	127
Índices del vector	0	1	2	3	4	5	6	...	114	115	116	117	118	119

- Para el segundo y siguientes elementos del vector elegidos al azar, sí hay que comprobar si el tipo de motor ya existe entre los ya elegidos. En caso de existir, deberá realizarse de nuevo la elección aleatoria, hasta que se obtenga un tipo de motor que no haya sido elegido hasta el momento.

Deberá realizar el programa atendiendo a las siguientes indicaciones:

- Se utilizarán las constantes simbólicas N y NUM_TIPO con valores 120 y 5, respectivamente. N es el número de códigos de piezas que existen y NUM_TIPO el número de tipos de motores.
- Se utilizará el vector v_codigos, de tipo entero y de tamaño N.
- Para cargar el vector v_codigos se llamará, desde el programa principal, a la función CargarVector(), **QUE NO HAY QUE CODIFICAR**, cuyo prototipo es:

```
void CargarVector(int vector_codigos[]);
```

- Una vez la función CargarVector() ha cargado el vector, en el programa principal será preciso generar números aleatorios. Para la generación de números aleatorios se utilizará la función rand(), que devuelve números aleatorios entre 0 y MAX_RANDOM, constante ya definida en C. Para el uso de la función rand() se deben incluir las librerías stdlib.h y time.h. Hay que generar la semilla para la función rand() mediante: srand((unsigned)time(NULL));
- En el ejemplo seguido, al tener NUM_TIPO valor 5, los 5 primeros elementos del vector v_codigos finalmente serán 5 códigos de piezas, donde cada una de ellas corresponde a un tipo de motor distinto. Por último, el programa mostrará por pantalla los NUM_TIPO de códigos de las piezas seleccionadas para hacer el control de calidad, pero en el programa pedido **NO HAY QUE CODIFICAR** esta funcionalidad.

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( iniciación, 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
```

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);
```