PROGRAMACIÓN

(GRADOS EN INGENIERO MECÁNICO, ELÉCTRICO, ELECTRÓNICO INDUSTRIAL y QUÍMICO INDUSTRIAL)

Sesión	13 (Estructuras de Datos: números aleatorios)	
Temporización	1 hora (no presencial)	
Objetivos formativos	• Conocer la sintaxis de C para la definición de nuevas tipologías de datos e implementación de estructuras de datos estáticas: "arrays" unidimensionales y multidimensionales, cadenas de caracteres y registros (con y sin parte variante). • Implementar programas modulares en lenguaje de programación C. Identificar y corregir errores sintácticos que surgen durante la codificación. • Resolver problemas sencillos con "arrays" unidimensionales y multidimensionales, aplicando las operaciones básicas sobre los mismos (acceso directo a elementos individuales y acceso secuencial). • Resolver ejercicios sencillos de cadenas de caracteres con representación semi-estática. Conocer las operaciones básicas que se realizan sobre las cadenas de caracteres. • Resolver (diseñar e implementar) ejercicios sencillos de registros con y sin parte variante: construir operaciones abstractas sobre tipos abstractos de datos. • Resolver (diseñar e implementar) problemas con modelos complejos de información: representar el modelo de información mediante tipologías básicas y constructores de tipos estructurados. Acceder a elementos individuales de información. • Conocer los algoritmos básicos de clasificación y búsqueda internas. Aplicarlos a la resolución (diseño e implementación) de sub-problemas de clasificación por diferentes criterios, con datos representados mediante vectores de registros. • *Utilizar números pseudo-aleatorios para problemas de simulación y juegos de azar.* • Diseñar e implementar programas que resuelven problemas de ingeniería usando operaciones abstractas sobre tipos abstractos de datos: representar el modelo de información mediante una combinación de estructuras de datos y construir operaciones complejas mediante técnicas de diseño modular y programación estructurada. • Probar con datos operacionales la correctitud de los módulos y programas desarrollados e identificar y corregir los errores lógicos que surjan.	
Competencias a desarrollar	• RD1: Poseer y comprender conocimientos • RD2: Aplicación de conocimientos • UAL1: Conocimientos básicos de la profesión • UAL3: Capacidad para resolver problemas • UAL6: Trabajo en equipo • FB3: Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en la ingeniería.	X X X X X

Materiales	Sesiones de teoría (grupo docente) 8,10,11 y 12 + Bibliografía Tema 4 + Internet IDEs : Dev-C++/Code::Blocks (freeware)	
Tarea	Desarrollar los tres programas propuestos en esta ficha de trabajo y presentar un informe según modelo que se adjunta.	
Fecha de entrega	Siguiente sesión del Grupo de Trabajo.	
Criterios de éxito	• Terminar en el tiempo previsto la tarea. • Demostrar, en una prueba escrita u oral, mediante las respuestas a las preguntas del profesor que ha alcanzado los objetivos formativos.	
Plan de trabajo	**Actividad**	**Temporización**
	Estudio de la sintaxis de C para generar números pseudo-aleatorios enteros y reales dentro de un determinado rango.	10 mn
	Diseño de los sub-algoritmos correspondientes a cada uno de los ejercicios propuestos. Nota: puede simultanear esta actividad con las dos siguientes (para cada ejercicio).	20 mn
	Implementación en lenguaje C de los sub-programas correspondientes a los algoritmos diseñados.	10 mn
	Pruebas: los programas desarrollados serán validados utilizando como mínimo los datos de prueba suministrados. Nota: en caso de detectar errores en esta fase de pruebas, estos deberán ser corregidos modificando el código fuente y/o el algoritmo correspondiente.	10 mn
	Elaboración de la documentación a presentar según modelo adjunto, así como de la respuesta a las cuestiones planteadas en el mismo.	10 mn

Sintaxis de C: Generación de números pseudo-aleatorios

Números aleatorios.

Un número aleatorio es un número cuyo valor no es conocido a priori de una forma determinista (es impredecible su valor). Los números aleatorios se aplican a una amplia variedad de problemas, que van desde los juegos de azar hasta sofisticadas técnicas de cálculo numérico, pasando por simulaciones, por generación de datos de entrada y valores de forma automática,... En las bibliotecas estándar de C ***stdlib.h*** y ***time.h*** se dispone de varias funciones para generar números aleatorios entre un rango determinado de valores.

- La función rand()

Genera y devuelve a través del identificador de la función un número aleatorio entero en el intervalo [0,RAND_MAX]. La constante RAND_MAX está definida en el archivo ***stdlib.h*** como 2E15-1. Ejemplo:

```
#include <stdlib.h>
int x;
. . .
x = rand();
```

Para generar números aleatorios enteros en un intervalo [0,n), o lo que es lo mismo en el intervalo [0,n-1], se procede de la siguiente manera:

```
x= rand() % n   /* n es una variable entera positiva */
```

Alternativamente, se puede usar la macro pre-definida ***random()***:

```
x = random(n);
```

Si queremos generar números aleatorios enteros comprendidos entre 1 y n (ambos incluidos) se procedería de la siguiente manera:

```
x = rand() %n +1;
```
o equivalentemente: `x= random(n) + 1;`

En general, para generar números aleatorios enteros entre a y b (a<b), procederíamos así:

`random(n)`	genera números aleatorios enteros en el intervalo [0,n-1]
`random(n) + a`	genera números en el intervalo [a,a+n-1]
	a+n-1=b → n=b-a+1
`random(b-a+1)+a`	genera números en el intervalo [a,b]=a,a+1,a+2,...,b-1,b

Para generar números aleatorios reales en un intervalo [a,b], se procede de la siguiente manera:

`1.0*rand()/RAND_MAX`	genera un número aleatorio real en el intervalo [0,1]
`n*1.0*rand()/RAND_MAX`	genera un número aleatorio real en el intervalo [0,n]
`a+n*1.0*rand()/RAND_MAX`	genera un número aleatorio real en el intervalo [a,a+n]
	a+n=b → n=b-a
`a+(b-a)*1.0*rand()/RAND_MAX`	genera número aleatorio real en [a,b]

- función srand()

Inicializa el generador de números aleatorios con un valor aleatorio obtenido del reloj del sistema. Se ejecuta solo una vez al principio del programa, consiguiendo de esta forma que en sucesivas ejecuciones del mismo programa no se genere la misma secuencia de números aleatorios.

```
#include <stdlib.h>
#include <time.h>
time_t t;
. . .
srand((unsigned) time(&t));
```

Equivalentemente, se puede usar la macro `randomize()`, pero hay que incluir también las dos bibliotecas anteriores.

Ejercicios: desarrollo de programas

Ejercicio 1

Construir un programa que simule el lanzamiento de dos monedas no trucadas un número prefijado de veces introducido por teclado, y que calcule e imprima en pantalla el porcentaje de veces que sale 0, 1 y 2 caras.

Datos de prueba

	Porcentaje de veces (%) que sale		
n	**0 caras**	**1 cara**	**2 caras**
10	30.00	50.00	20.00
100	25.00	55.00	20.00
1000	26.10	48.40	25.50
10000	25.40	49.79	24.81
100000	24.98	49.98	25.04
1000000	25.00	50.01	24.99

Ejercicio 2

Construir un programa que simule el lanzamiento de un dado no trucado un número prefijado de veces introducido por teclado, y que calcule e imprima en pantalla el porcentaje de veces que sale cada uno de los seis números del dado.

Datos de prueba

	Porcentaje de veces que sale cada número (%)					
n	**1**	**2**	**3**	**4**	**5**	**6**
10	0.00	20.00	0.00	40.00	20.00	20.00
100	19.00	22.00	15.00	21.00	11.00	12.00
1000	17.40	18.60	15.10	18.50	15.40	15.00
10000	16.88	17.17	17.22	16.80	16.29	15.64
100000	16.70	16.74	16.36	16.61	16.92	16.67
1000000	16.66	16.65	16.69	16.66	16.65	16.70

Ejercicio 3

Construir un programa que inicialice con valores aleatorios un vector de 20 fechas (día: 1-31, mes: 1-12, año: 2000-2006), lo clasifique por fecha de más reciente a menos reciente, e imprima en pantalla el vector original y el vector clasificado.

Datos de prueba

N=5

	1	**2**	**3**	**4**	**5**
Vector original:	1-2-2000	20-2-2004	12-10-2004	31-9-2001	9-3-2001
Vector ordenado:	1-2-2000	9-3-2001	31-9-2001	20-2-2004	12-10-2004
Vector orden inverso:	12-10-2004	20-2-2004	31-9-2001	9-3-2001	1-2-2000

Nota: los datos de prueba de los ejercicios anteriores se dan a modo ilustrativo, dado que se están generando números aleatorios que cambian de una ejecución a otra.

Asignatura	Programación		
Plan de Estudios	Grados en Ingeniero Mecánico, Eléctrico, Electrónico Industrial y Químico Industrial		
Actividad	Trabajo individual	**Sesión**	13
Tiempo empleado			

Apellidos, nombre	DNI	Firma

1.- Documentación del diseño y de la implementación de los programas desarrollados.

Documentar de forma adecuada los productos de ingeniería obtenidos en las fases principales de desarrollo de los sub-programas de esta práctica: para el diseño utilizar la notación formal de pseudo-código propuesta en las clases de teoría y para la codificación utilizar directamente el listado fuente en lenguaje C.	
Ejercicio	(enunciado del ejercicio)
Diseño preliminar	Diseño de datos: (nuevas tipologías) Estructura del programa: (diagrama de módulos) Interfaces entre módulos: (nombre + lista de parámetros formales)
Ejercicio	
Diseño preliminar	
. . . .	

2.- Resultados de aprendizaje (reflexión): marque con una cruz los objetivos que cree haber alcanzado tras realizar esta actividad, y rellene en el campo de observaciones aquellos aspectos que cree que necesita mejorar (si los hubiera):

Objetivos formativos	Cumplimiento
• Conocer la sintaxis de C para la definición de nuevas tipologías de datos e implementación de estructuras de datos estáticas: "arrays" unidimensionales y multidimensionales, cadenas de caracteres y registros (con y sin parte variante). • Implementar programas modulares en lenguaje de programación C. Identificar y corregir errores sintácticos que surgen durante la codificación. • Resolver problemas sencillos con "arrays" unidimensionales y multidimensionales, aplicando las operaciones básicas sobre los mismos (acceso directo a elementos individuales y acceso secuencial). • Resolver ejercicios sencillos de cadenas de caracteres con representación semi-estática. Conocer las operaciones básicas que se realizan sobre las cadenas de caracteres.	

• Resolver (diseñar e implementar) ejercicios sencillos de registros con y sin parte variante: construir operaciones abstractas sobre tipos abstractos de datos. • Resolver (diseñar e implementar) problemas con modelos complejos de información: representar el modelo de información mediante tipologías básicas y constructores de tipos estructurados. Acceder a elementos individuales de información. • Conocer los algoritmos básicos de clasificación y búsqueda internas. Aplicarlos a la resolución (diseño e implementación) de sub-problemas de clasificación por diferentes criterios, con datos representados mediante vectores de registros. • Utilizar números pseudo-aleatorios para problemas de simulación y juegos de azar. • Diseñar e implementar programas que resuelven problemas de ingeniería usando operaciones abstractas sobre tipos abstractos de datos: representar el modelo de información mediante una combinación de estructuras de datos y construir operaciones complejas mediante técnicas de diseño modular y programación estructurada. • Probar con datos operacionales la correctitud de los módulos y programas desarrollados e identificar y corregir los errores lógicos que surjan.	
Observaciones	