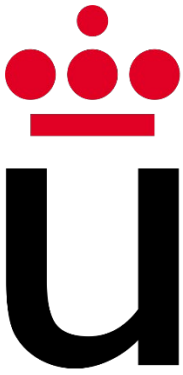


Universidad
Rey Juan Carlos

Ejercicios de la Asignatura

Introducción a la Programación

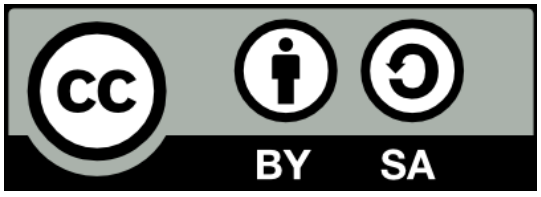
Curso 2024/2025

Grados de impartición:

- Grado en Ingeniería de Computadores
- Grado en Ingeniería de la Ciberseguridad

Autores:

Diego Hortelano, Gerardo Reyes, Manuel Rubio



Índice de contenidos

Tema 1: Introducción a la Programación. Ejercicios

Ejercicio 1

El máximo común divisor (MCD) de dos números enteros es el mayor número entero que divide a ambos de manera exacta. Por ejemplo, 6 es el MCD de 12 y 18, mientras que 16 es el MCD de 48 y 64. El antiguo matemático griego Euclides desarrolló un algoritmo para obtener el MCD basándose en el principio de que el máximo común divisor de dos números no cambia si el número más grande se reemplaza por su diferencia con el número más pequeño. Por ejemplo, si tomamos los números A = 24 y B = 18, podemos obtener su MCD de la siguiente forma:

1. Restamos el menor (B) al mayor (A):
    - A = 24 – B = 24 – 18 = 6
    - Ahora tenemos que A = 6 y B = 18.
2. Restamos el menor (A) al mayor (B):
    - B = 18 – A = 18 – 6 = 12
    - Ahora tenemos que A = 6 y B = 12.
3. De nuevo, restamos el menor (A) al mayor (B):
    - B = 12 – A = 12 – 6 = 6
    - Ahora tenemos que A = 6 y B = 6.
4. Como los dos números A y B son iguales, el MCD es 6.

Escribe un algoritmo que permita calcular el MCD de esta forma.

Ejercicio 2

Escribe el pseudocódigo para el ejercicio anterior.

Ejercicio 3

Escribe el pseudocódigo de un algoritmo que muestre el número con mayor valor de una lista dada.

Ejercicio 4

Escribe el pseudocódigo del algoritmo de ordenación burbuja detallado en el Tema 1.

Tema 2: Fundamentos del lenguaje C. Ejercicios

Ejercicio 1

Escribe un programa que lea un `int`, un `float` y un `char` y los escriba, en forma de "eco".

Ejercicio 2

Escribe un programa que lea y escriba una cadena de caracteres (definida como `char cadena [LENGTH]`, siendo `LENGTH` una constante) de tres formas diferentes. ¿Qué pasa si insertamos una palabra con más caracteres que LENGTH?

Ejercicio 3

Escribe un programa que lea un `char` y lo imprima como `int` para ver su código ASCII. Después, el programa deberá leer un entero e imprimir el carácter ASCII asociado a su valor.

Ejercicio 4

Implementa un programa que convierta los euros leídos a dólares, utilizando el cambio ($1€ = 1.11$\$). Utiliza constantes literales y simbólicas.

Ejercicio 5

Implementa un programa que lea un entero e imprima por pantalla el doble de ese entero.

Ejercicio 7

Implementa un programa que lea los datos necesarios para calcular el área de un triángulo e imprima dicha área.

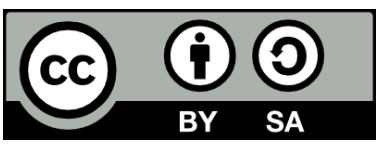
Ejercicio 6

Implementa un programa que lea el radio de un círculo y calcule e imprima su área. Utiliza constantes.

Ejercicio 9

Modifica el ejercicio 7 para que la salida sea la siguiente, utilizando un único `printf` (los números de altura y base tienen una anchura de 7 dígitos, de los cuales 2 son decimales):

```
Dame la base del triangulo:
12
Dame la altura del triangulo:
3
El area del triangulo con las siguientes caracteristicas:
        - Altura:    3.00
        - Base:   12.00
es: 18.00
Process finished with exit code 0
```



Tema 3: Operadores y Expresiones. Ejercicios

Ejercicio 1

Sean a, b, c, d, x, y, z variables enteras, ¿Qué valor proporcionarán las siguientes expresiones si a=4, b=5, c=4, d=0, e=8, x=1, y=1, z=1?

a) 0
b) 1
c) b==c
d) (a+2 < b) || (c < d-3)
e) (x <= y) && (y <= z)
f) ((!((a==6) && (b<=3))) || (c <= 7))

Ejercicio 2

Sea a=3, b=4, c=5, ¿cómo se evalúan estas expresiones?

a) d = (a<2) ? 2 : 4
b) d = (a++ <= 3) ? 2: 4
c) d = (++a <=3) ? 2 : 4
d) d = (a == b)
e) d = (a != b) || (a == c)
f) d = ((a*b) > c)
g) d = (b != c) && (a != c) && (b != c) || (a != a)
h) d = (c % b) < a
i) d = (a *= b) > c ? 2 : 1

Ejercicio 3

En el siguiente programa se utilizan los operadores lógicos binarios (™™ y N¶y se pide al usuario que introduzca tres números (num7?num8.y.num9). Analiza el código, poniendo especial atención en el uso de los operadores lógicos y a partir de tu análisis, responde a las preguntas que se encuentran en los recuadros en gris del código.

```c
#include <stdio.h>

int main() {
 int num1, num2, num3;
 // Solicitar tres números
 printf("Introduce el primer número: ");
 scanf("%d", &num1);
 printf("Introduce el segundo número: ");
 scanf("%d", &num2);
 printf("Introduce el tercer número: ");
 scanf("%d", &num3);
 //  1) ¿Qué hace la siguiente línea de código?
 if (num1 % 2 == 0 && num2 % 2 == 0 && num3 % 2 == 0) {
   printf("2) ¿Qué mensaje debe mostrarse si se cumple esta condición? \n");
 else {
   printf("3) ¿Qué mensaje debe mostrarse si no se cumple la condición?\n");
 }
 return 0;
}
```

Ejercicio 4

El siguiente programa utiliza el operador relacional (~), así como el operador ternario (?¿). En este programa, se pide al usuario que introduzca dos números (num7.y.num8) e imprime el valor almacenado en resultado. Analiza el código, poniendo especial atención en el uso de los operadores mencionados y a partir de tu análisis, rellena la tabla de abajo.

```c
#include <stdio.h>

int main() {
    int num1, num2, resultado;
    // Solicitar dos números:
    printf("Introduce el primer número: ");
    scanf("%d", &num1);
    printf("Introduce el segundo número: ");
    scanf("%d", &num2);

    // uso  del operador ternario ?:
    resultado = (num1 > num2) ? num1 : num2;

    // realiza una acción a partir de la comparación
    printf("El valor de resultado es: %d\n", resultado);
    return 0;
}
```

Rellena la tabla siguiente:

num1	num2	resultado
1	2	
2	1	
	3	3
		5
4	4	
6		*6*

Ejercicio 5

Utilizando el operador ternario (?:) y los operadores necesarios, escribe programas que:

1. Lea dos números e imprima el menor de ellos.
2. Lea un número e imprima si dicho número es par o impar.
3. Lea un número e imprima su valor absoluto.

Ejercicio 6

El siguiente programa solicita al usuario un número entero (entero) y un número decimal (decimal). El programa realiza una operación de suma entre ambos números (>) y almacena el resultado en la variable resultado. Analiza la conversión de tipos realizada por el compilador. A partir de tu análisis, responde a las preguntas.

```c
#include <stdio.h>

int main() {
    int entero;
    float decimal;
    float resultado;

    // Solicitar el número entero
    printf("Introduce un número entero: ");
    scanf("%d", &entero);

    // Solicitar el número decimal
    printf("Introduce un número decimal: ");
    scanf("%f", &decimal);

    // Realizar una operación que involucre ambos tipos (int y
float)
    resultado = entero + decimal;

    // Mostrar el resultado
    printf("El resultado de sumar %d y %.2f es: %.2f\n", entero,
decimal, resultado);
    return 0;
}
```

Preguntas:

1) ¿Se trata de una conversión implícita o explícita?
2) ¿La variable resultado tendrá un valor entero o en punto flotante?

Ejercicio 7

El siguiente programa solicita al usuario un número entero (entero) y un número flotante (decimal). El programa realiza una división entre ambos números (/) y almacena el resultado en la variable resultado. Analiza la conversión de tipos realizada por el compilador. A partir de tu análisis, responde a las preguntas.

```c
#include <stdio.h>

int main() {
    float decimal;
    int entero;
    float resultado;

    // Solicitar el número decimal
    printf("Introduce un número decimal: ");
    scanf("%f", &decimal);

    // Solicitar el número entero
    printf("Introduce un número entero: ");
    scanf("%d", &entero);

    // Realiza una división, convierte los tipos y almacena en
resultado
    resultado = decimal / (float)entero;

    // Mostrar el resultado
    printf("El resultado de dividir %.2f entre %d es: %.2f\n",
decimal, entero, resultado);
    return 0;
}
```

Preguntas:

1) ¿Se trata de una conversión implícita o explícita?
2) ¿La variable resultado tendrá un valor entero o flotante?
3) ¿Qué sucedería si cambiamos la línea de código marcada en negro por:

   resultado = decimal + entero;

Ejercicio 8

Escribe un programa que, haciendo uso de la biblioteca math.h, lea un número y muestre por pantalla su raíz cuadrada. Imprime el resultado como un entero.

Ejercicio 9

Escribe un programa que, haciendo uso de la biblioteca math.h, lea una base y un exponente, y muestre por pantalla el resultado de elevar dicha base al exponente. Muestra el resultado con 3 decimales.

Ejercicio 10

El siguiente código muestra dos operadores bit a bit de C. Ejecuta el código y comprueba qué resultados obtiene. Realiza pruebas con otros valores. ¿A qué se debe? ¿Cómo funcionan estos operadores? ¿Qué otros operadores bit a bit existen en C?

```c
#include <stdio.h>

int main() {
    int x = 8;
    int resultado1 = x << 2;
    int resultado2 = x >> 1;
    printf("Valor de resultado1: %d\n Valor de resultado2: %d\n",
           resultado1, resultado2);
    return 0;
}
```

Tema 4: Estructuras de Control. Ejercicios I

Ejercicio 1

Escribe un programa que pida al usuario un número. Si el número es par, el programa deberá imprimir por pantalla "El número es par", mientras que si es impar deberá escribir "El número es impar". Utiliza únicamente estructuras `if`.

Ejercicio 2

Modifica el ejercicio anterior (paridad de un número) para que utilice el esquema `if-else`.

Ejercicio 3

Escribe un programa que pida al usuario dos números enteros para realizar una división. Si el divisor es diferente de cero, realizará la división y mostrará el resultado por pantalla. Si no, mostrará por pantalla un mensaje de error indicando que no se puede dividir por cero.

Ejercicio 4

Implementa un programa que indique si una persona es alta o no en función de su altura utilizando estructuras `if-else` anidadas siguiendo las siguientes indicaciones:

- Si la persona mide menos de 1.50, el programa mostrará un mensaje diciendo que es una persona baja.
- Si la persona mide menos de 1.70, el programa mostrará un mensaje diciendo que es una persona media.
- Si la persona mide menos de 1.90, el programa mostrará un mensaje diciendo que es una persona alta.
- Si la persona mide más de 1.90, el programa mostrará un mensaje diciendo que es una persona muy alta.

Ejercicio 5

Implementa un programa que pida al usuario un año y calcule si es bisiesto o no. Un año es bisiesto si es mayor a 1582 (antes no existían bisiestos) y es divisible entre 4. Además, si es también divisible entre 100 debe serlo a su vez entre 400.

Ejercicio 6

Implementa un programa que pida una letra y calcule si es vocal mayúscula, vocal minúscula o no es vocal.

Ejercicio 7

Implementa un programa que pida al usuario 3 números enteros, y calcule y escriba por pantalla el mayor de ellos.

Ejercicio 8

Utilizando la estructura `switch-case`, implementa un programa que pida por teclado el idioma del usuario (1-Inglés, 2-Español, etc.) y después salude al usuario en función de su idioma (Good morning, Buenos días, etc.).

Ejercicio 9

Implementa un programa que calcule la nota final que vas a obtener en Introducción a la programación en función de la nota obtenida en las cuatro pruebas.

- Si la nota es menor que 5 será SUSPENSO
- Si está entre 5 y 6.9 será APROBADO.
- Si está entre 7 y 8.9 será NOTABLE.
- Si está entre 9 y 9.9 será SOBRESALIENTE.
- Si es 10 será MATRÍCULA DE HONOR.

Ejercicio 10

Implementa un programa que pida al usuario un número del 0 al 9 e imprima la tabla de multiplicar de ese número.

Ejercicio 11

Implementa un programa que pida números enteros por teclado al usuario y los sume hasta que el usuario introduzca un 0.

Ejercicio 12

Implementa un programa que calcule la media de n números reales introducidos por teclado. Inicialmente se pedirá que el usuario introduzca el número de números que va a introducir después.

Ejercicio 13

Implementa un programa que calcule el factorial de un número pedido al usuario.

Ejercicio 14

Implementa un programa que use un bucle para calcular la potencia de un número, pidiendo al usuario la base y el exponente.

Ejercicio 15

Implementa un programa que pida al usuario un número entero entre 0 y 100. Si este número es menor que 0 o mayor que 100, debemos indicarlo y volver a pedir el número. Si el número se encuentra en el intervalo correcto, se deben mostrar todos los números pares entre el número y 100.

Ejercicio 16

Implementa un programa que indique si un número introducido por teclado es primo o no.

Ejercicio 17

Implementa un programa que calcule la media de varios números. Estos números serán solicitados al usuario, hasta que introduzca un 0.

Ejercicio 18

Implementa un programa que muestre todos los números del 1 al 100 en una tabla de 10x10, como se muestra a continuación.

```
1       2       3       4       5       6       7       8       9       10
11      12      13      14      15      16      17      18      19      20
21      22      23      24      25      26      27      28      29      30
31      32      33      34      35      36      37      38      39      40
41      42      43      44      45      46      47      48      49      50
51      52      53      54      55      56      57      58      59      60
61      62      63      64      65      66      67      68      69      70
71      72      73      74      75      76      77      78      79      80
81      82      83      84      85      86      87      88      89      90
91      92      93      94      95      96      97      98      99      100

Process finished with exit code 0
```

Tema 4: Estructuras de Control. Ejercicios II

Ejercicio 1

Crea una calculadora que realice operaciones de `suma`, `resta`, `multiplicación` y `división`. Para ello el programa debe leer el primer operando, el operador y el segundo operando. A continuación, nuestro programa debe calcular y mostrar el resultado obtenido.

Ejercicio 2

Escribe un programa que pida al usuario dos números e imprima todos los que hay entre dichos números, incluyendo ambos números.

Ejercicio 3

Escribe un programa que pida al usuario un número `entero` entre `0` y `100`. Si este número es menor que `0` o mayor que `100`, debemos indicarlo y volver a pedir el número. Si el número se encuentra en el intervalo correcto, se deben mostrar todos los números pares entre el número dado y `100`.

Ejercicio 4

Implementa un programa que muestre el primer y último dígitos de un número entero. Por ejemplo, para el número `20331`, muestre el `2` y el `1`.

Ejercicio 5

Implementa un programa que intercambie el primer y el último dígito de un número entero. Por ejemplo, dado el `24781` obtenga el `14782`.

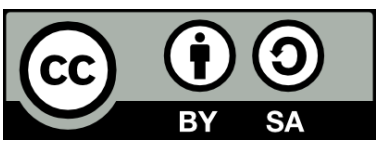
Ejercicio 6

Implementa un programa que, dado un número entero, cree otro número entero con las cifras del primero en orden inverso. Por ejemplo, dado el `123` obtenga el `321`.

Ejercicio 7

Implementa un programa en C que genere un número aleatorio entre `0` y `50`. Después, pedirá números al usuario hasta que éste lo adivine, indicándole si el número leído es correcto, menor o mayor que el generado aleatoriamente. Para obtener un número aleatorio puede utilizarse la función `rand()` de la biblioteca `stdlib.h`:

https://cplusplus.com/reference/cstdlib/rand/



Ejercicio 8

Implementa un programa que, dado un número `n` entero positivo, calcule el siguiente sumatorio:

$$S = \sum_{i=1}^{n} i$$

Ejercicio 9

Implementa un programa que, dado un número `n` entero positivo, calcule el siguiente sumatorio:

$$S = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (i+j)^2$$

Por ejemplo, el resultado para un número `n=5` es `390`, mientras que para un número `n=1` es `0`.

Ejercicio 10

Implementa un programa que pida al usuario dos números `A` y `B` y que imprima todos los múltiplos de `13` que haya entre ellos (incluyendo `A` y `B`). Nuestro programa debe comprobar que `A` y `B` son mayores que `0`, y, además, que `B > A`. Si no se cumple alguna de estas condiciones, nuestro programa debe volver a solicitar los números.

Ejercicio 11

Implementa un programa que realice una multiplicación de dos números pedidos al usuario mediante la suma. Por ejemplo, la multiplicación `2x3` se realizaría con `2+2+2`.

Ejercicio 12

Implementa un programa que imprima los números enteros entre `A y B` (asignados en el código como `200` y `350`), contando de `m` en `m` (siendo `m` solicitado al usuario). Nuestro programa debe comprobar que `m` es mayor que `1` (y volver a solicitarlo en caso de que no se cumplan las condiciones).

Ejercicio 13

Implementa un programa que imprima por pantalla un tablero de ajedrez de 8x8 utilizando dos bucles `for`, uno para las filas y otro para las columnas. Para representar el tablero se utilizará el carácter `'N'` para las casillas negras y el carácter `'B'` para las casillas blancas. La primera casilla puede ser blanca o negra. Ten en cuenta que la suma de la fila y columna de una casilla de un color es par y la del otro color es impar. ¿Podrías hacerlo con un operador condicional ternario (`? :`)?

Ejercicio 14

Implementa un programa que pida al usuario un número `n` con decimales y muestre por pantalla ese número, su mitad, la mitad de su mitad, etc., es decir `n`, `n/2`, `n/4` (o `n/2/2`)... el programa terminará cuando el resultado sea menor que `1`.

Tema 5: Arrays y Cadenas de Caracteres.
Ejercicios I

Ejercicio 1

Crea un array de 10 posiciones, guardando en cada posición el índice de la posición multiplicado por 10. Recorre el array imprimiendo por pantalla su contenido.

Ejercicio 2

Escribe un programa que lea de teclado las temperaturas medias de los 12 meses del año y calcule la temperatura media anual.

Ejercicio 3

Implementa un programa que recorra un array (puede estar inicializado en nuestro propio programa) y muestre por pantalla su elemento mayor.

Ejercicio 4

Escribe un programa que declare un array de 10 elementos. El programa pedirá al usuario el número de elementos a almacenar (siempre menor que 10), y después pedirá cada uno de los elementos. Finalmente, imprime por pantalla únicamente los elementos que insertados en el array.

Ejercicio 5

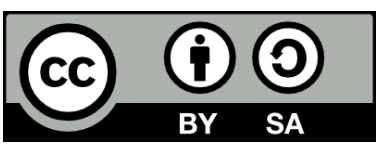
Escribe un programa que declare un array de 10 elementos, donde cada posición indique la nota media del alumno con número de expediente igual a la posición (dicho número será un número entero del 0 al 9). Pide al usuario por teclado que indique cuántas posiciones va a rellenar, indicando en cada paso la posición y la nota media. Finalmente, imprime por pantalla las notas medias de cada expediente introducido junto a su número de expediente (únicamente aquellas notas ya introducidas en el array).

Ejercicio 6

Escribe un programa que, dado un array, copie sus elementos en otro array, imprimiendo después ambos arrays para comprobar si la copia es correcta.

Ejercicio 7

Escribe un programa que, dados dos arrays, imprima por pantalla si dichos arrays son o no iguales.



Ejercicio 8

Escribe un programa que, dado un array, cree un array inverso, es decir, con los elementos almacenados en el orden opuesto, imprimiendo después ambos arrays para comprobar si la nuestro programa funciona correctamente.

Ejercicio 9

Escribe un programa que, dados dos arrays, obtenga un array suma de los dos dados, imprimiendo después los tres arrays.

Ejercicio 10

Escribe un programa que, dado un array, ordene de menor a mayor sus elementos.

Ejercicio 11

Escribe un programa que, pida a un usuario introducir los valores (con decimales) de una matriz (array bidimensional) 4x3. Después, el programa mostrará los valores máximos de cada fila y de cada columna, indicando en cada caso la fila/columna a la que pertenece.

Ejercicio 12

Escribe un programa que, dada una matriz (array bidimensional), obtenga su matriz traspuesta.

Ejercicio 13

Escribe un programa que, dadas dos matrices (arrays bidimensionales), obtenga una matriz suma de las mismas.

Ejercicio 14

Escribe un programa que, dadas dos matrices (arrays bidimensionales), obtenga una matriz multiplicación de las mismas.

Ejercicio 15

Escribe un programa que permita leer una cadena de caracteres por teclado, imprima el número de caracteres que tiene dicha cadena y finalmente imprima dicha cadena al revés.

Ejercicio 16

Escribe un programa que lea una cadena de caracteres y lo imprima por pantalla cambiando las letras mayúsculas por minúsculas y las minúsculas por mayúsculas.

Ejercicio 17

Implementa un programa que, dada una cadena de caracteres, la copie en otra cadena de caracteres, de manera similar a como lo hace la función `strcpy` de la biblioteca `string.h`.

Ejercicio 18

Implementa un programa que, dadas dos cadenas de caracteres, indique si son iguales o si una es mayor que otra, de manera similar a como lo hace la función `strcmp` de la biblioteca `string.h`.

Ejercicio 19

Implementa un programa que, dadas dos cadenas de caracteres, concatene una de ellas en la otra, de manera similar a como lo hace la función `strcat` de la biblioteca `string.h`.

Tema 5: Arrays y Cadenas de Caracteres. Ejercicios II

Ejercicio 1

Desarrolla un programa en C que pida al usuario un número `n` (entre 2 y 20) y después pida `n` enteros, almacenándolos en un array. A continuación, nuestro programa debe calcular la suma de todos los números pares del array, mostrando el resultado.

Ejercicio 2

Crea un programa en C que cuente cuántas veces aparece un número específico en un array de enteros.

Ejercicio 3

Partiendo del programa creado en el ejercicio anterior, modifícalo para que muestre por pantalla en qué posiciones se encuentra el entero buscado. Las posiciones deben almacenarse en otro array e imprimirse tras completar la búsqueda.

Ejercicio 4

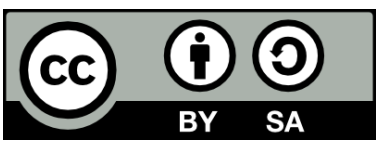
Implementa un programa que invierta un array sin utilizar un array auxiliar (sí puede utilizarse una variable auxiliar). No es suficiente con imprimir el array, el array original debe quedar invertido en memoria.

Ejercicio 5

Implementa un programa que, dados dos números enteros `n1` y `n2`, almacene en un array todos los números impares que se encuentren entre ellos (incluyendo dichos números). Además, obtén el resultado de multiplicar todos ellos.

Ejercicio 6

Implementa un programa en C que, dado un array, ordene de menor a mayor sus elementos. Puedes utilizar diferentes algoritmos de ordenación.



Ejercicio 7

Implementa un programa que solicite al usuario un número entero `n` entre 3 y 100 (ambos incluidos), y que repita esta operación hasta que el usuario inserte un número correcto. Después, guarda en un array los `n` primeros dígitos de la sucesión de Fibonacci. Finalmente, imprime este array indicando su posición y valor. Por ejemplo: Término 1: 0; Término 2: 1; Término 3: 1; Término 4: 2; ...

Ejercicio 8

Escribe un programa en C que, pida a un usuario introducir los valores (con decimales) de una matriz 4x3. Después, el programa mostrará los valores máximos de cada fila y de cada columna, indicando en cada caso la fila/columna a la que pertenece.

Ejercicio 9

Crea un programa que encuentre la diagonal principal y la diagonal secundaria de una matriz cuadrada, almacenando cada una de ellas en un array.

Ejercicio 10

Implementa un programa en C que lea una cadena de caracteres y muestre por pantalla la longitud de la misma, de manera similar a como lo hace la función `strlen` de `string.h`.

Ejercicio 11

Implementa un programa que lea una cadena de caracteres y elimine todos los espacios en blanco de dicha cadena. El programa debe eliminar dichos espacios de la cadena original.

Ejercicio 12

Desarrolla un programa que cuente la cantidad de letras (mayúsculas o minúsculas), números y símbolos de una cadena de caracteres.

Tema 6: Punteros. Ejercicios

Ejercicio 1

Implementa un programa que pida por teclado un número entero y lo almacene en la variable n. A continuación, declara un puntero p que apunte a la variable n, y la modifique para que tome el valor 3. Finalmente, incrementa el valor de la variable n en una unidad e imprime por consola cuánto vale n y qué valor tiene el dato apuntado por p.

Ejercicio 2

Implementa un programa que inicialice un vector y después lo recorra utilizando punteros.

Ejercicio 3

Utilizando punteros para recorrer arrays, pide al usuario un array a y crea un array b que contenga los mismos elementos que a pero en orden inverso.

Ejercicio 4

Implementa un programa que inicialice un matriz y después muestre sus elementos de tres formas diferentes, utilizando punteros y aritmética de punteros.

Ejercicio 5

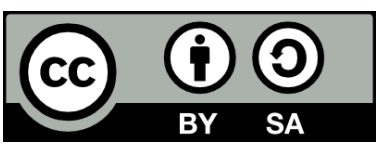
Crea un programa que declare e inicialice una variable de cada uno de los tipos básicos de C, así como un puntero a cada una de dichas variables y un puntero a cada uno de dichos punteros. Finalmente, el programa deberá imprimir la dirección y valor de cada uno de las variables y punteros.

Ejercicio 6

Dado un array, imprime la dirección de memoria de cada una de sus posiciones y su valor, de dos formas diferentes (utilizando arrays y punteros).

Ejercicio 7

Implementa un programa que sume todos los elementos de un array utilizando punteros.



Ejercicio 8

Implementa un programa que busque un determinado elemento en un array utilizando punteros, almacenando la posición que ocupa dicho elemento en el array, o -1 si no se encuentra. Crea también un puntero a dicho elemento, que apunte a NULL si el elemento no está. Imprime la posición y la dirección de memoria del elemento.

Ejercicio 9

Un número capicúa es aquel que se lee igual de izquierda a derecha que de derecha a izquierda (por ejemplo, el 1441). De igual manera, podríamos extrapolar este término a nuestros arrays, si pueden leerse de igual manera en ambos sentidos. Implementa un programa que indique si un array dado es o no capicúa.

Ejercicio 10

Implementa un programa que sume dos matrices (arrays bidimensionales) utilizando punteros.

Ejercicio 11

Implementa un programa que obtenga la matriz traspuesta de una matriz dada (array bidimensional) utilizando punteros.

Ejercicio 12

Implementa un programa que calcule la longitud de una cadena de caracteres utilizando un array.

Ejercicio 13

Implementa un programa que concatene dos cadenas de caracteres utilizando punteros.

Tema 7: Funciones. Ejercicios I

Ejercicio 1

Implementa un programa que calcule el área de un círculo. El cálculo del área debe estar definido en una función.

Ejercicio 2

Implementa un programa que pida al usuario un número n y calcule la suma de todos los números de la serie $n + (n-1) + (n-2) + \cdots + 1$. La serie debe estar definida en una función.

Ejercicio 3

Implementa un programa que pida al usuario tres enteros (día, mes y año). La petición de datos debe estar en una función, que además debe devolver un 1 si la fecha es válida o un 0 en caso contrario, además del día, mes y año leídos. La fecha se considera válida si el día está en el rango [0, 31], el mes [0, 12] y el año [0, 2024].

Ejercicio 4

Implementa un programa que contenga una función para buscar un número dentro de un array. La función devolverá la posición del array que contiene dicho número o -1 en caso de no encontrarlo.

Ejercicio 5

Implementa una función para imprimir por pantalla el contenido de una matriz que se recibe como parámetro.

Ejercicio 6

Implementa un programa que calcule la suma de dos números n y m a través de la suma lenta (recursivo). En esta suma, un número se va incrementando de uno en uno, mientras que el otro se decrementa al mismo ritmo, hasta que el número decrementado llegue a 0:

n	m
5	3
6	2
7	1
8	0

Ejercicio 7

Implementa un programa que pida al usuario un número n y calcule la suma de todos los números de la serie n + (n−1) + (n−2) ... 1. En este caso el cálculo de la serie debe estar definido en una función que obtenga el resultado de manera recursiva.

Ejercicio 8

Implementa un programa que reciba en la función `main` tres argumentos: el nombre del usuario, su año de nacimiento y su nota media. A continuación, el programa deberá imprimir por pantalla el mensaje: "Hola `NOMBRE`, tienes `AÑOS` anyos. Tu nota media es de `NOTA`".

Tema 7: Funciones. Ejercicios II

Ejercicio 1

Implementa un programa en C que convierta un número decimal en un número binario utilizando una función.

Ejercicio 2

Implementa un programa en C que indique si un número es primo o no utilizando una función, la cual debe retornar un booleano (o la aproximación de C para este tipo de datos).

Ejercicio 3

Implementa un programa en C que muestre el elemento más grande de un array utilizando una función.

Ejercicio 4

En las olimpiadas se miden las marcas de los atletas en segundos porque resultan más apropiados para almacenarlos en un soporte informático. Sin embargo, resulta muy incómodo para los aficionados leer las marcas en segundos. Por esta razón, nos han pedido que implementemos un programa que tenga una función que pase un tiempo expresado en segundos a su equivalente en horas, minutos y segundos.

La lectura del tiempo en segundos y la escritura del tiempo en horas, minutos y segundos debe realizarse en la función main, mientras que la conversión debe estar en una función.

Por ejemplo, si se introduce 4550 segundos, nuestro programa debe mostrar: 1 hora, 15 minutos y 50 segundos.

Tema 7: Funciones. Ejercicios III (Recursividad)

Ejercicio 1

Escribe una función recursiva que ordene de menor a mayor un array de enteros basándote en la siguiente idea: coloca el elemento más pequeño en la primera ubicación, y luego ordene el resto del arreglo con una llamada recursiva.

Ejercicio 2

Implementa una función recursiva que calcule el $n-ésimo$ término de la sucesión de Fibonacci. Recuerda que la sucesión de Fibonacci se define como $F(0) = 0, F(1) = 1$, y $F(n) = F(n-1) + F(n-2)$ para $n > 1$.

Ejercicio 3

Crea una función recursiva que determine si una cadena de caracteres es palíndroma. Una cadena es palíndroma si se lee igual de izquierda a derecha que de derecha a izquierda. Por ejemplo, "ana" y "$reconocer$" son palíndromas.

Ejercicio 4

Diseña una función recursiva que imprima los dígitos de un número entero positivo en orden inverso. Por ejemplo, si el número es 1234, la función debe imprimir 4321.

Ejercicio 5

Escribe una función recursiva que devuelva el máximo común divisor de dos números enteros positivos a y b. El máximo común divisor de dos números se define como el mayor número que divide exactamente a ambos números. Puedes usar el algoritmo de Euclides, que dice que $mcd(a,b) = mcd(b, a\ \%\ b)$ si $b\ != 0$, y $mcd(a,0) = a$.

Ejercicio 6

Diseña e implementa una función recursiva que devuelva el número de ocurrencias de un carácter c en una cadena de caracteres s. Por ejemplo, si $c = 'a'$ y $s = "casa"$, la función debe retornar 2.

Tema 8: Memoria Dinámica. Ejercicios

Ejercicio 1

Implementa un programa que lea un entero en una función creado con memoria dinámica y lo devuelva al programa principal, que lo imprimirá por pantalla.

Ejercicio 2

Implementa una función que reciba un entero *n* y retorne un array con los *n* primeros enteros.

Ejercicio 3

Implementa otra función que reciba el array creado en la función anterior y su tamaño, y retorne la suma de todos los elementos del array.

Ejercicio 4

Implementa un programa que cree de manera dinámica una matriz de nxn y calcule la suma de todos los elementos almacenados en ella. Para ello, implementa funciones para:

1. Reservar espacio para una matriz, conociendo sus dimensiones.
2. Inicializar la matriz con números aleatorios.
3. Calcular la suma de sus elementos.
4. Imprimir la matriz.
5. Muestra en la función main el resultado de la suma.

Ejercicio 5

Implementa un programa que cree un array en memoria dinámica con espacio suficiente para 30 variables enteras.

A continuación, el programa deberá almacenar números leídos por teclado, bien hasta alcanzar el espacio máximo del array o bien hasta que el usuario introduzca un 0. Esta solicitud debe realizarse en una función.

Tras ello, nuestro programa deberá liberar la memoria no utilizada (posiciones a partir del elemento 0, incluyendo este). Esta operación deberá realizarse en otra función, la cual retornará la longitud del array tras el procesamiento llevado a cabo.

Ejercicio 6

Implementa un programa que solicite al usuario un entero n, y cree un array en memoria dinámica con espacio para n números decimales. A continuación, el programa deberá recorrer el array y obtener el producto de todos los números almacenados. Este cálculo debe realizarse en una función.

Ejercicio 7

Implementa un programa que cree un array en memoria dinámica con 50 posiciones y las inicialice a 0. A continuación, el usuario podrá seleccionar una posición y un valor para que sea almacenado en esa posición del array. Este proceso se repetirá hasta que el usuario seleccione una posición no válida (menor que 0 o mayor que 50). Ten en cuenta que las posiciones pueden no ser continuas.

Tras esto, nuestro programa deberá mover todos los elementos diferentes de 0 a las primeras posiciones, manteniendo el orden, y reducir el tamaño del array en memoria. Este procesamiento se realizará en una función, que retornará la nueva longitud del array.

Ejercicio 8

Crea un programa que lea por pantalla una cadena de caracteres de hasta 100 caracteres y la almacene en memoria dinámica. Después, ajusta el tamaño en memoria de la cadena de caracteres al mínimo necesario (teniendo en cuenta el carácter nulo).

Ejercicio 9

Continuando con el ejercicio anterior, crea otra cadena en memoria dinámica que contenga el mensaje cifrado utilizando el cifrado César con desplazamiento 6. Este cifrado debe realizarse en una función. Ten en cuenta que este cifrado solamente contendrá letras, no pudiendo contener otros símbolos.

Ejercicio 10

En relación con el ejercicio anterior, crea una nueva función que, dada una cadena cifrada, retorne la cadena descifrada, retornando un puntero a memoria dinámica.

Ejercicio 11

Implementa un programa que cree, inicialice e imprima una matriz dinámica con 5 filas, donde cada fila tendrá un número de columnas diferente. La matriz debe contener estos valores:

1	2	3	4	5
6	7	8	9	
10	11	12		
13	14			
15				

Tema 9: Estructuras y Tipos de Datos Enumerados. Ejercicios

Ejercicio 1

Define una estructura que almacene una fecha con día, mes y año.

Ejercicio 2

Define una estructura que guarde el nombre de un alumno, su fecha de nacimiento y número de expediente.

Ejercicio 3

Define una estructura que almacene las coordenadas de un punto 3D (x,y,z). Después, crea una variable de tipo punto 3D y asígnale valores. Finalmente, muestra el valor de cada campo por consola.

Ejercicio 4

Define una estructura que permita almacenar el nombre, precio y cantidad disponible de un producto. Después, nuestro programa deberá pedir al usuario el número total de productos a almacenar, creando un array de dicho número de elementos. Implementa también una función que solicite al usuario información sobre cada producto y la almacene en el array.

Ejercicio 5

Define un enumerado que represente los días de la semana: el lunes debe tener asignado el valor 1, hasta el domingo que tendrá valor 7. Utiliza esos datos para validar datos que el usuario ingrese por teclado.

Tema 9: Estructuras y Tipos de Datos Enumerados. Ejercicios II

Ejercicio 1

Vamos a implementar un videojuego de rol. En este videojuego, existen los siguientes elementos:

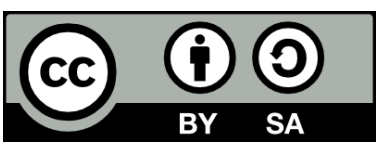
- Armas, cuyas características son: nombre (será suficiente con 50 caracteres), tipo (que debe codificar si es una espada, una lanza, un escudo, un arco), daño (un número entero) y defensa (otro número entero).
- Hechizos, con las siguientes características: nombre (50 caracteres máximo), tipo (fuego, tierra, viento, agua), daño (un número entero) y sanación (otro entero).
- Personajes, cuyas características son: nombre (100 caracteres máximo), tipo (luchador, mago, sanador, arquero), daño base (un número entero), defensa base (otro entero), vida (también entero), armas (hasta un total de 5), y hechizos (hasta un total de 3).
- Finalmente, el jugador llevará un equipo de hasta 5 personajes.

Implementa las siguientes funciones:

- Una función que, dado un personaje, retorne su arma más poderosa (con más ataque).
- Una función que, dado un personaje, retorne su arma de tipo escudo con mayor defensa.
- Una función que, dado un equipo, retorne el arma más poderosa y el nombre del personaje con dicha arma.
- Una función que, dado un equipo y un tipo de personaje, retorne el hechizo con mayor sanación y el nombre del personaje con ese hechizo.
- Una función que, dado un equipo y un tipo de hechizo, retorne el hechizo con mayor ataque y el nombre del personaje con dicho hechizo.

Implementa una función `main` que permita probar las funciones implementadas. Toda lectura y escritura debe realizarse en la función `main`.

Un **posible ejemplo de entrada** de nuestro programa es el siguiente: en primer lugar, un número entero, que indica el número de personajes que contiene nuestro equipo. A continuación, cada uno de los personajes. La primera línea de cada personaje contiene una cadena de caracteres con su nombre; la segunda línea contiene un carácter con su tipo (`L` – luchador, `M` – mago, `S` – Sanador y `A` – Arquero); la tercera línea contiene 3 enteros, correspondientes al ataque, defensa y vida del personaje. A continuación, se incluyen las armas y los hechizos, ocupando en ambos casos dos líneas. La separación entre las armas de un personaje y sus hechizos es un `*`. Esta misma separación (`*`) nos indica también que los hechizos de un personaje han terminado, comenzando un nuevo personaje.



Las armas y los hechizos vienen representados en dos líneas. La primera de ellas es una cadena de caracteres con el nombre del arma o hechizo. La segunda contiene un carácter y dos enteros. El carácter indica el tipo (`E` – Espada, `L` – Lanza, `S` – Escudo, `A` – Arco para las armas y `F` – Fuego, `T` – Tierra, `V` – Viento, `A` – Agua para los hechizos). Los dos enteros indican el daño y la defensa en el caso de las armas, y el daño y la sanación en el caso de los hechizos.

Una posible entrada siguiendo estas directrices puede ser:

```
5
Aragorn
L
34 23 420
Anduril
E 140 0
Eski
S 10 40
Arco1
A 34 12
*
Bola de fuego
F 100 0
Susurro de aire
V 20 0
*
Legolas
A
40 15 300
Espada doble
E 30 0
Arco elfo
A 100 0
Arco de montaraz
A 35 0
Escudo Orco
S 5 25
*
Sana sanita
A 0 10
Flechas de llamas
F 120 0
*
Gimli
L
30 50 640
Maza enana
E 60 0
Escudo enano
S 5 60
Escudo roto
S 0 15
Ballesta enana
A 30 0
*
Terremoto
T 50 0
*
Gandalf el blanco
```

```
M
40 20 380
Espada blanca
E 30 0
Vara de mago
L 20 20
*
Destello
F 70 0
Sanacion Blanca
A 0 40
*
Gandalf el gris
M
20 15 360
Espada normal
E 20 0
Vara
L 10 20
*
Luz mágica
F 20 0
No pasaran
T 120 0
Sanacion
A 0 40
*
```

Por ejemplo, si utilizamos las funciones implementadas con los siguientes parámetros:

- Obtener el arma más poderosa del personaje almacenado en la posición 1 (Legolas), considerando que comienzan en 0.
- Obtener el escudo con mayor defensa del personaje almacenado en la posición 2 (Gimli), considerando que comienzan en 0.
- Obtener el arma más poderosa de todos los personajes del equipo, así como el nombre del personaje que la posee.
- Obtener el hechizo con mayor sanación del equipo de aquellos personajes que sean de tipo MAGO, así como el nombre del personaje que lo tiene.
- Obtener el hechizo con mayor daño del equipo de tipo FUEGO, así como el nombre del personaje que lo tiene.

Obtendríamos la siguiente salida:

```
Arma mas poderosa del personaje en la posicion 1:
Arma: Arco elfo, de tipo ARCO. Dano: 100; Defensa: 0.
Mejor escudo del personaje en la posicion 2:
Arma: Escudo enano, de tipo ESCUDO. Dano: 5; Defensa: 60.
El personaje con el arma mas poderosa es: Aragorn. El arma es:
Arma: Anduril, de tipo ESPADA. Dano: 140; Defensa: 0.
El personaje con el hechizo con mas sanacion es: Gandalf el blanco.
El hechizo es:
Hechizo: Sanacion Blanca, de tipo AGUA. Dano: 0; Sanacion: 40.
El personaje con el hechizo de FUEGO con mas dano es: Legolas. El
hechizo es:
Hechizo: Flechas de llamas, de tipo FUEGO. Dano: 120; Sanacion: 0.
```

Tema 10: Ficheros. Ejercicios

Ejercicio 1

Implementa un programa que escriba en un fichero de texto el valor de una variable leída de teclado y su cuadrado. Por ejemplo: "El cuadrado de 4 es 16".

Ejercicio 2

Dado un fichero de texto que almacena un float en cada línea, implementa un programa que lea todos los datos del fichero y calcule la suma de los mismos, mostrando el resultado por pantalla.

Ejercicio 3

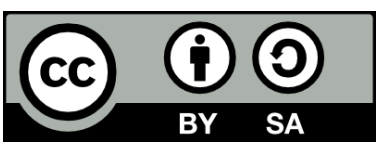
Implementa un programa que almacene en un fichero binario un valor entero. A continuación, el programa deberá leer ese valor e imprimirlo por pantalla. Abre el fichero creado con un bloc de notas. ¿Qué contiene?

Ejercicio 4

Implementa un programa que almacene en un fichero binario un array de floats. A continuación, deberá leer ese array e imprimirlo por pantalla. Abre el fichero binario creado en esta ocasión. ¿Cuál es su contenido?

Ejercicio 5

Implementa un programa que almacene en un fichero binario una estructura st_alumno que se compone de nombre, edad y nota media. Después, lee esa estructura e imprime cada campo por pantalla. ¿Qué contiene el fichero en esta ocasión?



Práctica 1 - No evaluable

Introducción a la Programación

Índice

1. Categorías de calificaciones

1.1. Introducción

En este ejercicio el objetivo es emplear una sentencia `switch`.

1.2. Enunciado del problema

Implementa un programa que calcule una nota final como una media ponderada de cuatro pruebas, y escriba la calificación según las categorías tradicionales (suspenso, aprobado, notable, sobresaliente y matrícula de honor).

En primer lugar el programa leerá las cuatro notas n_1, n_2, n_3 y n_4 por teclado. Sus valores serán números reales en el intervalo $[0, 10]$. No es necesario comprobar esto, se asume que son precondiciones que van a cumplirse necesariamente. Posteriormente se calculará la media ponderada de las notas:

$$m = n_1 \cdot 0{,}1 + n_2 \cdot 0{,}1 + n_3 \cdot 0{,}2 + n_4 \cdot 0{,}6$$

Finalmente se imprimirá imprima por pantalla un texto $T(m)$ correspondiente a la calificación según:

$$T(m) = \begin{cases} \text{“}\texttt{SUSPENSO}\text{”} & \text{si } 0 \leqslant m < 5 \\ \text{“}\texttt{APROBADO}\text{”} & \text{si } 5 \leqslant m < 7 \\ \text{“}\texttt{NOTABLE}\text{”} & \text{si } 7 \leqslant m < 9 \\ \text{“}\texttt{SOBRESALIENTE}\text{”} & \text{si } 9 \leqslant m < 10 \\ \text{“}\texttt{MATRICULA DE HONOR}\text{”} & \text{si } m = 10 \end{cases}$$

Nota: Aunque también se puede implementar utilizando sentencias `if`, se pide expresamente usar una sentencia `switch`.

1.2.1. Descripción de la entrada

La entrada contiene una línea con las notas n_i, para i=1,...,4, separadas por espacios en blanco. Al final de la última nota habrá un salto de línea.

1.2.2. Descripción de la salida

La salida contendrá el texto $T(m)$, seguido de un salto de línea.

1.2.3. Ejemplo de entrada

```
7.3 7 6.5 8.22↵
```

1.2.4. Salida para el ejemplo de entrada

```
NOTABLE↵
```

2. Media aritmética

2.1. Introducción

La media aritmética de un conjunto de valores numéricos es posiblemente su "estadístico" más importante. En este ejercicio calcularéis al media de un conjunto de números reales.

2.2. Enunciado del problema

Se pide implementar un programa que, dado un número entero positivo $n > 0$, lea n números reales $x_i \in \mathbb{R}$, para $i = 1, \ldots, n$, y calcule su media aritmética. Formalmente se trata de hallar:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

2.2.1. Descripción de la entrada

La entrada contiene, en su primera línea n, seguido de un salto de línea. La segunda línea contendrá la secuencia de números reales $\langle x_1, x_2, \ldots, x_n \rangle$, separados por espacios en blanco, y acabará en un salto de línea.

2.2.2. Descripción de la salida

La salida contendrá el valor real $\bar{x}$, que contendrá **exactamente 3 cifras decimales**, seguido de un salto de línea.

2.2.3. Ejemplo de entrada

```
5↵
3.41112 -1.222 6.53 4.2332 -0.34↵
```

2.2.4. Salida para el ejemplo de entrada

```
2.522↵
```

3. Factorial

3.1. Introducción

El factorial de un número se puede calcular de varias maneras. En este ejercicio usaréis un bucle.

3.2. Enunciado del problema

Se pide implementar un programa que, dado un número entero no negativo $n \geqslant 0$, calcule el factorial de ese número, es decir:

$$n! = 1 \cdot 2 \cdot 3 \cdot \ \cdots \ \cdot (n-1) \cdot n = \prod_{i=1}^{n} i$$

Se recuerda que $0! = 1$.

3.2.1. Descripción de la entrada

La entrada contiene el número n, seguido de un salto de línea.

3.2.2. Descripción de la salida

La salida contendrá $n!$, seguido de un salto de línea.

3.2.3. Ejemplo de entrada

```
5↵
```

3.2.4. Salida para el ejemplo de entrada

```
120↵
```

4. Potencia

4.1. Introducción

En este ejercicio volveréis a usar un bucle para calcular una potencia.

4.2. Enunciado del problema

Se pide implementar un programa que, dado un número real b, y un número entero n, calcule la potencia b^n:

4.2.1. Descripción de la entrada

La entrada contiene el número b, un espacio en blanco, n, y un salto de línea.

4.2.2. Descripción de la salida

La salida contendrá b^n, con **exactamente 3 cifras decimales**, seguido de un salto de línea.

4.2.3. Ejemplo de entrada

```
3.2 4↵
```

4.2.4. Salida para el ejemplo de entrada

```
104.858↵
```

5. Secuencia de pares

5.1. Introducción

En este ejercicio volveréis a usar bucles para imprimir una secuencia de números pares

5.2. Enunciado del problema

Se pide implementar un programa que, dado número entero n entre 0 y 100, imprima todos los números pares desde n hasta 100. El programa probará si $n \in [0, 100]$. Si esto no ocurre deberá seguir leyendo enteros por teclado hasta que sí se cumple la condición.

5.2.1. Descripción de la entrada

La entrada podrá contener varias líneas. En cada una habrá un número entero seguido de un salto de línea. Todas las líneas contendrán un número fuera del intervalo [0,100], excepto la última, que contendrá $n \in [0, 100]$. seguido de un salto de línea.

5.2.2. Descripción de la salida

La salida contendrá una línea con todos los números pares desde n hasta 100, en orden creciente, separados por espacios en blanco. La línea terminará con un salto de línea.

5.2.3. Ejemplo de entrada

```
-7↵
127↵
87↵
```

5.2.4. Salida para el ejemplo de entrada

```
88 90 92 94 96 98 100↵
```

6. ¿Es primo?

6.1. Introducción

En este ejercicio implementaréis un programa para determinar si un número entero es primo.

6.2. Enunciado del problema

Se pide implementar un programa que, dado número positivo n, determine si es o no primo. Un número primo solo es divisible por él mismo y por 1. El 1 no se considera número primo.

6.2.1. Descripción de la entrada

La entrada contendrá n seguido de un salto de línea

6.2.2. Descripción de la salida

Si n es primo la salida será el texto "Si", mientras que será "No" si no lo es. La línea terminará con un salto de línea.

6.2.3. Ejemplo de entrada

```
23↵
```

6.2.4. Salida para el ejemplo de entrada

```
Si↵
```

7. Media aritmética - variante

7.1. Introducción

En este ejercicio calcularéis al media de un conjunto de n números enteros, cuando *a priori* no se conoce n.

7.2. Enunciado del problema

Se pide implementar un programa que lea por teclado $n + 1$ números enteros. Los primeros n serán distintos de 0 y el último será 0. Es decir, el programa dejará de leer enteros cuando el último introducido sea precisamente un 0. EL programa hallará e imprimirá la media de los primeros n números introducidos.

7.2.1. Descripción de la entrada

La entrada contiene una línea con $n+1$ enteros, separados por espacios en blanco, donde solo el último será igual a 0. La línea terminará con un salto de línea.

7.2.2. Descripción de la salida

La salida contendrá el valor real de la media de los n enteros distintos de 0 introducidos por teclado. Se escribirá con **exactamente 3 cifras decimales**, y estará seguido de un salto de línea.

7.2.3. Ejemplo de entrada

```
3 6 17 8 0↵
```

7.2.4. Salida para el ejemplo de entrada

```
8.500↵
```

8. Tabla 10 x 10

8.1. Introducción

En este ejercicio tendréis que usar dos bucles anidados.

8.2. Enunciado del problema

Se pide implementar un programa que imprima los números del 1 al 100 de la siguiente manera (los números están separados por tabuladores (\t)):

```
1   2   3   4   5   6   7   8   9   10
11  12  13  14  15  16  17  18  19  20
21  22  23  24  25  26  27  28  29  30
31  32  33  34  35  36  37  38  39  40
41  42  43  44  45  46  47  48  49  50
51  52  53  54  55  56  57  58  59  60
61  62  63  64  65  66  67  68  69  70
71  72  73  74  75  76  77  78  79  80
81  82  83  84  85  86  87  88  89  90
91  92  93  94  95  96  97  98  99  100
```

8.2.1. Descripción de la entrada

No hay entrada para este programa

8.2.2. Descripción de la salida

La tabla, como se especifica en el enunciado.

8.2.3. Ejemplo de entrada

8.2.4. Salida para el ejemplo de entrada

```
1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10↵
11→ 12→ 13→ 14→ 15→ 16→ 17→ 18→ 19→ 20↵
21→ 22→ 23→ 24→ 25→ 26→ 27→ 28→ 29→ 30↵
31→ 32→ 33→ 34→ 35→ 36→ 37→ 38→ 39→ 40↵
41→ 42→ 43→ 44→ 45→ 46→ 47→ 48→ 49→ 50↵
51→ 52→ 53→ 54→ 55→ 56→ 57→ 58→ 59→ 60↵
61→ 62→ 63→ 64→ 65→ 66→ 67→ 68→ 69→ 70↵
71→ 72→ 73→ 74→ 75→ 76→ 77→ 78→ 79→ 80↵
81→ 82→ 83→ 84→ 85→ 86→ 87→ 88→ 89→ 90↵
91→ 92→ 93→ 94→ 95→ 96→ 97→ 98→ 99→ 100↵
```

Práctica 2 - No evaluable

Introducción a la Programación

Índice

1. Códigos ASCII

1.1. Introducción

En Pascal un carácter se representa en binario mediante una secuencia de unos y ceros (un byte), que también puede ser interpretada como un número al que se denomina código ASCII. En este ejercicio generaréis códigos ASCII a partir de caracteres.

1.2. Enunciado del problema

Se pide implementar un programa que, dado un carácter c, determine su correspondiente código ASCII.

1.2.1. Descripción de la entrada

La entrada consistirá en el carácter c, seguido de un salto de línea.

1.2.2. Descripción de la salida

La salida contendrá el código ASCII de c, seguido de un salto de línea.

1.2.3. Ejemplo de entrada

```
a↵
```

1.2.4. Salida para el ejemplo de entrada

```
97↵
```

2. Función techo

2.1. Introducción

La función techo es aquella que, dado un número real x, calcula el número entero más próximo a x por exceso, es decir, el menor número entero igual o mayor que x. Se suele denotar como $\lceil x \rceil$. En este ejemplo tendréis que usar la instrucción IF, junto con la función matemática trunc.

2.2. Enunciado del problema

Se pide implementar un programa que, dado $x \in \mathbb{R}$, calcule el entero $\lceil x \rceil$.

2.2.1. Descripción de la entrada

La entrada contiene x seguido de un salto de línea.

2.2.2. Descripción de la salida

La salida contendrá el entero $\lceil x \rceil$, seguido de un salto de línea.

2.2.3. Ejemplo de entrada

```
-4.7↵
```

2.2.4. Salida para el ejemplo de entrada

```
-4↵
```

3. Clasificación según ganancias

3.1. Introducción

En este ejercicio el objetivo es emplear sentencias IF en cascada.

3.2. Enunciado del problema

Se pide implementar un programa que, dada una cantidad (real) de euros $x \geqslant 0$ correspondiente a las ganancias de una persona al mes, imprima una palabra p por pantalla según las siguientes reglas:

$$p = \begin{cases} \text{“}\texttt{Pobre}\text{”} & \text{si } 0 \leqslant x < 645{,}3 \\ \text{“}\texttt{Interprofesional}\text{”} & \text{si } x = 645{,}3 \\ \text{“}\texttt{Mileurista}\text{”} & \text{si } 645{,}3 < x \leqslant 1200 \\ \text{“}\texttt{Privilegiado}\text{”} & \text{si } 1200 < x \leqslant 2000 \\ \text{“}\texttt{Rico}\text{”} & \text{si } 2000 < x < 6000 \\ \text{“}\texttt{Millonario}\text{”} & \text{si } 6000 \leqslant x \end{cases}$$

Nota: Es importante tener en cuenta que el salario mínimo interprofesional para 2013 es un número real que no puede expresarse de forma precisa en binario. Además, se debe seguir estrictamente los límites indicados.

3.2.1. Descripción de la entrada

La entrada contiene x seguido de un salto de línea.

3.2.2. Descripción de la salida

La salida contendrá la palabra p, seguida de un salto de línea.

3.2.3. Ejemplo de entrada

```
645.3↵
```

3.2.4. Salida para el ejemplo de entrada

```
Interprofesional↵
```

4. Resolución de ecuaciones de segundo grado

4.1. Introducción

En este ejercicio el objetivo es emplear sentencias IF en cascada, junto con bloques de código delimitados por limitadores BEGIN y END.

4.2. Enunciado del problema

Se pide implementar un programa que resuelva ecuaciones de segundo grado: $ax^2 + bx + c = 0$, con coeficientes $a, b, c \in \mathbb{R}$, con $a \neq 0$, y variable $x \in \mathbb{R}$. Tras leer los coeficientes, que definen la ecuación, se imprimirán por pantalla las soluciones de la ecuación (raíces de $ax^2 + bx + c$), definidas mediante:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

4.2.1. Descripción de la entrada

La entrada contiene una linea con los coeficientes a, b, y c, separados mediante espacios en blanco. La línea termina en un salto de línea.

4.2.2. Descripción de la salida

Las salidas por pantalla serán diferentes teniendo en cuenta las siguientes situaciones:

- Raíces distintas y reales. Se imprimirá x_1 en una primera línea y x_2 en la segunda línea (que acabará en un salto de línea).
- Raíces iguales. Se imprimirá x_1, seguida de un espacio en blanco y el texto "`(doble)`". Finalmente se escribirá un salto de línea.
- Raíces imaginarias. En este caso las raíces se pueden descomponer en una parte real (r) y otra imaginaria (s): $x_1 = r + si$, y $x_2 = r - si$. De esta manera, la salida tendrá en su primera línea r, un espacio en blanco, un símbolo $+$, un espacio en blanco, s, la letra `i`, y un salto de línea. La segunda línea tendrá r, un espacio en blanco, un símbolo $-$, un espacio en blanco, s, la letra `i`, y un salto de línea.

Todos los números reales se escribirán con 3 decimales exactamente.

4.2.3. Ejemplo de entrada 1

```
1 0 -1↵
```

4.2.4. Salida para el ejemplo de entrada 1

```
1.000↵
-1.000↵
```

4.2.5. Ejemplo de entrada 2

```
1 2 1↵
```

4.2.6. Salida para el ejemplo de entrada 2

```
-1.000 (doble)↵
```

4.2.7. Ejemplo de entrada 3

```
1 3 3↵
```

4.2.8. Salida para el ejemplo de entrada 3

```
-1.500 + 0.866i↵
-1.500 - 0.866i↵
```

5. Secuencias de caracteres

5.1. Introducción

El siguiente ejercicio consiste en generar secuencias de caracteres, y admite numerosas implementaciones válidas basadas en bucles. Algunas usan índices o contadores enteros, pero en Pascal también se pueden usar índices y contadores de tipo carácter, lo cual puede facilitar las implementaciones.

5.2. Enunciado del problema

Se pide implementar un programa que, dado dos caracteres c_1 y c_2, imprima por pantalla todos los caracteres de la tabla ASCII entre c_1 y c_2, ambos incluidos, y en orden (de menor a mayor código ASCII). Si el código ASCII de c_1 es mayor que el de c_2, no se imprimirá ningún carácter.

5.2.1. Descripción de la entrada

La primera línea de la entrada contiene el carácter c_1, seguido de un salto de linea. La segunda línea contiene el carácter c_2, seguido de un salto de linea.

5.2.2. Descripción de la salida

Se escribirá cada carácter de la secuencia seguido de un salto de línea.

5.2.3. Ejemplo de entrada

```
8↵
C↵
```

5.2.4. Salida para el ejemplo de entrada

```
8↵
9↵
:↵
;↵
<↵
=↵
>↵
?↵
@↵
A↵
B↵
C↵
```

6. Inserta dígito en número ordenado

6.1. Introducción

Este ejercicio trata de realizar operaciones básicas con números enteros (no se pueden usar strings, arrays, listas, etc.).

6.2. Enunciado del problema

Considera un número entero positivo a cuyos dígitos aparecen ordenados orden creciente desde el más significativo hasta el menos (por ejemplo, $a = 245778$). Dado un dígito x, implementa una función que devuelva un nuevo número resultante de insertar x en a, de manera que los dígitos también queden ordenados en orden creciente. Ejemplos:

- $a = 245778$, $x = 0 \longrightarrow 245778$
- $a = 245778$, $x = 1 \longrightarrow 1245778$
- $a = 245778$, $x = 6 \longrightarrow 2456778$
- $a = 245778$, $x = 9 \longrightarrow 2457789$

6.2.1. Descripción de la entrada

En la primera línea se especifica a, seguido de un salto de línea. En la segunda línea se especifica x, seguido de un salto de línea.

6.2.2. Descripción de la salida

La salida contiene el nuevo número "ordenado", seguido de un salto de línea.

6.2.3. Ejemplo de entrada

```
245778↵
6↵
```

6.2.4. Salida para el ejemplo de entrada

```
2456778↵
```

7. Invertir número

7.1. Introducción

Este ejercicio también se propone para realizar operaciones básicas con números enteros (no se pueden usar strings, arrays, listas, etc.).

7.2. Enunciado del problema

Considera un número entero positivo de n dígitos: $x = d_{n-1}d_{n-2}\dots d_1d_0$, donde cada $0 \leqslant d_i \leqslant 9$ es un dígito decimal, y donde $d_{n-1} \neq 0$. Se pide implementar un programa que lea x por teclado, e lo imprima por pantalla invirtiendo el orden de sus dígitos: $d_0d_1\dots d_{n-2}d_{n-1}$, teniendo en cuenta que el dígito más significativo que se imprima debe ser distinto de 0.

7.2.1. Descripción de la entrada

Contendrá simplemente el entero x seguido de un salto de línea.

7.2.2. Descripción de la salida

La salida contiene el nuevo número "invertido", seguido de un salto de línea.

7.2.3. Ejemplo de entrada

253020↵

7.2.4. Salida para el ejemplo de entrada

20352↵

8. Pirámide

8.1. Introducción

Este ejercicio se propone para que implementéis un algoritmo con dos bucles anidados.

8.2. Enunciado del problema

Dado un número entero positivo n se pide imprimir por pantalla una pirámide de n filas con la siguiente estructura (para $n = 4$):

```
   X
  XXX
 XXXXX
XXXXXXX
```

8.2.1. Descripción de la entrada

Contendrá simplemente el entero n seguido de un salto de línea.

8.2.2. Descripción de la salida

La salida contendrá la pirámide. En la primera línea habrá $n - 1$ espacios en blanco, una 'X' y un salto de línea. La segunda línea tendrá $n - 2$ espacios en blanco, tres 'X' y un salto de línea. Este patrón continuará, hasta escribir la última fila, que no tendrá espacios, tendrá $2n - 1$ 'X', y un salto de línea.

8.2.3. Ejemplo de entrada

```
5↵
```

8.2.4. Salida para el ejemplo de entrada

```
␣␣␣␣X↵
␣␣␣XXX↵
␣␣XXXXX↵
␣XXXXXXX↵
XXXXXXXXX↵
```

Universidad Rey Juan Carlos

Práctica 1 - Evaluable

Introducción a la Programación

10 % de la nota final

Índice

1. Números primos (50 %)

1.1. Introducción

El objetivo de este ejercicio consiste en hallar números primos. Si se desea, se pueden usar arrays para implementar algoritmos eficientes, pero no son obligatorios. Naturalmente no podrá utilizarse ningún algoritmo visto en clase.

1.2. Enunciado del problema

Se pide implementar un programa que, dado un número entero positivo $n \in [1, 100]$, escriba por pantalla los primeros n números primos. Se recuerda que el 1 no se considera número primo. El primer número primo es el 2.

1.2.1. Descripción de la entrada

La entrada contiene el número n, seguido de un salto de línea.

1.2.2. Descripción de la salida

La salida contendrá n líneas, donde la i-ésima contendrá el número primo i-ésimo, seguido de un salto de línea.

1.2.3. Ejemplo de entrada

```
7↵
```

1.2.4. Salida para el ejemplo de entrada

```
2↵
3↵
5↵
7↵
11↵
13↵
17↵
```

2. Matriz simétrica (50 %)

2.1. Introducción

En este ejercicio tendréis que usar arrays bidimensionales (es decir, matrices) y usar los bucles adecuados para implementar un algoritmo **eficiente**.

2.2. Enunciado del problema

Se pide implementar un programa que, dado un número entero positivo $n \in [1, 10]$, lea una matriz $\mathbf{M}$ de enteros de dimensiones $n \times n$, y posteriormente indique si es o no simétrica. Si denominamos $m_{i,j}$ al elemento de la fila i y la columna j, una matriz simétrica es aquella que cumple $m_{i,j} = m_{j,i}$, para todo i y j.

2.2.1. Descripción de la entrada

La primera línea de la entrada contiene el número n, seguido de un salto de línea. A continuación la entrada contiene los elementos de $\mathbf{M}$, separados por espacios blancos, y ordenados por filas. Es decir, primero se especifican los elementos de la primera fila, de izquierda a derecha, luego de la segunda, etc. Después del último elemento habrá un salto de línea.

2.2.2. Descripción de la salida

Si $\mathbf{M}$ es simétrica se escribirá por pantalla "`Es simetrica`". En caso contrario se escribirá "`No es simetrica`". En ambos casos se escribirá un salto de línea después del texto (que no contiene tildes).

2.2.3. Ejemplo de entrada 1

```
3↵
3 1 2↵
1 8 4↵
2 4 7↵
```

2.2.4. Salida para el ejemplo de entrada 1

```
Es simetrica↵
```

2.2.5. Ejemplo de entrada 2

```
3↵
3 1 2↵
1 8 6↵
2 4 7↵
```

2.2.6. Salida para el ejemplo de entrada 2

```
No es simetrica↵
```

Universidad Rey Juan Carlos

Práctica 2 - Evaluable

Introducción a la Programación

10 % de la nota final

Índice

1. Variante de ordenación (50 %)

1.1. Introducción

En este ejercicio tendréis que implementar un algoritmo de ordenación de un array, e introducir una variante para modificar un segundo array.

1.2. Enunciado del problema

Se pide implementar un programa que, dado un número entero positivo $n \in [1, 100]$, lea un array de n enteros **v** (se puede asumir que todos sus elementos son diferentes), y posteriormente lea una cadena **c** de exactamente n caracteres (se puede asumir que no contiene espacios en blanco). El objetivo consiste en ordenar el array **v**, y aplicar la misma permutación necesaria para llevar a cabo la ordenación, pero sobre los caracteres de **c**. EL programa deberá imprimir por pantalla cómo queda la cadena de caracteres reorganizada de esta manera.

Por ejemplo, considerad $\mathbf{v} = [3, 7, 4, 8]$ y **c**="hola". Para ordenar **v** habría que intercambiar los dos elementos centrales. Si aplicamos ese intercambio a **c** obtendríamos "hloa", y el programa debe imprimir por pantalla esa cadena de caracteres.

Los programas usarán **memoria dinámica** para almacenar tanto **v** como **c**.

1.2.1. Descripción de la entrada

La entrada contiene el número n, seguido de un salto de línea. En la segunda línea habrá n números enteros, separados por espacios, correspondientes a los elementos de **v**. La tercera línea contendrá la cadena de caracteres **c**.

1.2.2. Descripción de la salida

La salida contendrá la cadena de caracteres reorganizada, seguida de un salto de línea.

1.2.3. Ejemplo de entrada

```
5↵
2 5 4 3 1↵
while↵
```

1.2.4. Salida para el ejemplo de entrada

```
ewlih↵
```

2. Convolución (50 %)

2.1. Introducción

La convolución es una operación matemática de enorme importancia, con aplicaciones en tratamiento de imágenes y sonido, o técnicas de inteligencia artificial. En este ejercicio tendréis que implementar una convolución de dos arrays de la misma longitud.

2.2. Enunciado del problema

Se pide implementar un programa que, dado un número entero positivo $n \in [1, 100]$, lea dos arrays de enteros u y v, de n elementos, y calcule su convolución (que se escribe como $\mathbf{u} * \mathbf{v}$). La convolución pedida en este caso es otro array $\mathbf{w}$ de enteros y de longitud $2n - 1$. Matemáticamente, el k-ésimo elemento de $\mathbf{w}$ se puede definir como:

$$w_k = (\mathbf{u} * \mathbf{v})_n = \sum_{\substack{i,j \\ i+j=k}} u_i v_j$$

asumiendo que los índices de los arrays empiezan en 0, como en C. De manera más explícita, los elementos de $\mathbf{w}$ se definen como:

$$\begin{aligned}
w_0 &= u_0 v_0 \\
w_1 &= u_0 v_1 + u_1 v_0 \\
w_2 &= u_0 v_2 + u_1 v_1 + u_2 v_0 \\
&\vdots \\
w_{n-2} &= u_0 v_{n-2} + u_1 v_{n-3} + \cdots + u_{n-3} v_1 + u_{n-2} v_0 \\
w_{n-1} &= u_0 v_{n-1} + u_1 v_{n-2} + \cdots + u_{n-3} v_2 + u_{n-2} v_1 + u_{n-1} v_0 \\
w_n &= u_1 v_{n-1} + u_2 v_{n-2} + \cdots + u_{n-2} v_2 + u_{n-1} v_1 \\
&\vdots \\
w_{2n-4} &= u_{n-3} v_{n-1} + u_{n-2} v_{n-2} + u_{n-1} v_{n-3} \\
w_{2n-3} &= u_{n-2} v_{n-1} + u_{n-1} v_{n-2} \\
w_{2n-2} &= u_{n-1} v_{n-1}
\end{aligned}$$

Observad que en una fila, si tenemos w_k, entonces los índices de cada pareja de términos multiplicándose suman exactamente k.

En este ejercicio todos los arrays se declararán usando **memoria dinámica**.

2.2.1. Descripción de la entrada

La primera línea de la entrada contiene el número n, seguido de un salto de línea. La segunda y tercera línea contendrá los elementos de $\mathbf{u}$ y $\mathbf{v}$, respectivamente,

separados por espacios blancos.

2.2.2. Descripción de la salida

Consistirá en una línea con los elementos de **w** separados por espacios. Después del último elemento se escribirá un salto de línea.

2.2.3. Ejemplo de entrada

```
3↵
2 4 1↵
5 2 3↵
```

2.2.4. Salida para el ejemplo de entrada

```
10 24 19 14 3↵
```

Universidad Rey Juan Carlos

Práctica 3 - Evaluable

Introducción a la Programación

20 % de la nota final

Índice

1. Números enteros muy grandes

1.1. Introducción

El objetivo de esta práctica consiste en desarrollar una calculadora de números enteros "muy grandes".

1.2. Enunciado del problema

Se pide implementar un programa que realice una serie de operaciones aritméticas con dos números enteros muy grandes x e y, que puedan contener hasta 50 dígitos. Éstos se especificarán de la misma manera que los números enteros habituales (una serie de dígitos, o un signo menos seguido de una secuencia de dígitos).

1.2.1. Descripción de la entrada

La primera línea contiene un carácter que indica la operación a realizar. Los caracteres posibles, junto con la operación a realizar, son:

- '`+`' : suma
- '`-`' : resta
- '`*`' : multiplicación

La segunda línea contiene x, y la tercera y. Todas las líneas terminan en un salto de línea.

1.2.2. Descripción de la salida

La salida contiene una línea con el resultado (un número entero muy grande), seguido de un salto de línea, según la operación a realizar:

- $x + y$
- $x - y$
- $x \cdot y$

Si el resultado contiene más de 50 dígitos se escribirá la frase "`DESBORDAMIENTO`"

1.2.3. Ejemplos de entrada y salida

En esta práctica se suministran los casos de prueba que se probarán en el DOMjudge, que contienen varios ejemplos de entradas y sus correspondientes salidas.

Ejercicio 1

Escribe un programa en C que lea un número entero *n*, el cual debe estar entre 5 y 20. En caso de no cumplirse esta condición se pedirá al usuario que introduzca un nuevo número. Esto se repetirá hasta que el usuario introduzca un número válido.

Después, el programa leerá *n* números enteros positivos, los cuales almacenará en un array. Podemos asumir que todos los números introducidos serán mayores que cero, por lo que no es necesario realizar ninguna comprobación. Por último, el programa deberá imprimir los *n/2* (división entera) menores números del array.

Para obtener la máxima nota se debe hacer uso de funciones y memoria dinámica. Se valorará también la eficiencia del programa.

Como ejemplo, se incluyen las siguientes entradas y salidas:

Ejemplo de entrada	**Ejemplo de salida**
10 2 5 78 32 12 9 15 12 8 49	2 5 8 12 12
3 32 7 83 71 23 92 32 72 12	12 23 32

Nótese que en el segundo ejemplo de entrada los primeros dos enteros corresponden a números de elementos no válidos (no están entre 5 y 20).

Ejercicio 2

Escribe una **única** función en C que permita obtener el número de letras mayúsculas y el número de letras minúsculas que contiene una cadena de caracteres, la cual puede contener cualquier carácter. Incluye su invocación en el siguiente código:

```c
#include <stdio.h>

// Incluye aquí la función pedida

int main() {
    int nMayus;
    int nMinus;
    char cadena[50] = "Esto ES Una PosiBLE cadena!! nO??";


    // Incluye la llamada a la función, así como otras sentencias si las necesitas

    printf("El número de letras mayúsculas es: %d\n", nMayus);
    printf("El número de letras minúsculas es: %d\n", nMinus);
```

Aunque no es necesario su uso, se incluye la siguiente información sobre los caracteres ASCII:

Carácter	Valor ASCII
'A'	65
'Z'	90

Carácter	Valor ASCII
'a'	97
'z'	122

Ejercicio 3

Se pide implementar una función **recursiva** para multiplicar dos enteros no negativos a y b. Es decir, se pide hallar el producto $a \cdot b = a + a + \cdots + a$ $(b\ veces)$. También sería válido el producto $a \cdot b = b + b + \cdots + b$ $(a\ veces)$.

No es necesario que la función sea eficiente, pero obsérvese que sólo se permite sumar (no se puede usar el operador *).

Ejercicio 4

Hacienda necesita desarrollar una aplicación para controlar ciertas empresas. De cada empresa es necesario almacenar su nombre (hasta 50 caracteres), el nombre de su director (hasta 70 caracteres) y el tipo de sociedad (Limitada, Anónima o Cooperativa). Además, cada empresa puede tener hasta 5 propiedades (pueden ser menos), las cuales vienen dadas por la calle donde se encuentran (hasta 50 caracteres), código postal (un entero) y los metros cuadrados (un número decimal) que tiene la propiedad.

Nuestra aplicación debe poder almacenar al menos 20 empresas diferentes.

A continuación, debemos escribir funciones que permitan:

1. Añadir una nueva empresa, con todas sus propiedades. En este caso, al ser nuestra aplicación podremos elegir cómo el usuario realiza la entrada de nuestro programa. Por ello, es necesario especificar cómo se introduciría por teclado una nueva empresa (con todos los datos necesarios), incluyendo un ejemplo para que el usuario no tenga ningún problema al introducir los datos. No es necesario pedir los datos uno a uno al usuario, podemos asumir que los introducirá siguiendo el ejemplo dado. El programa propuesto debe realizar la lectura respetando la entrada propuesta. Añadir una nueva empresa no debe eliminar ninguna de las ya existentes. En caso de intentar introducir más empresas de las soportadas por nuestro programa, se debe mostrar un aviso y no almacenar ningún dato.
2. Obtener el número de empresas que tenemos almacenadas de un determinado tipo de sociedad (Limitada, Anónima o Cooperativa), el cual debe pasarse como parámetro.
3. Obtener el total de metros cuadrados que tienen las propiedades de un director determinado (dado su nombre). Ten en cuenta que una misma persona podría dirigir más de una empresa.

Crea una pequeña función main que, haciendo uso de las funciones anteriores:

- Almacene 20 empresas pedidas al usuario.
- Obtenga el número de empresas que sean del tipo de sociedad Anónima.
- Obtenga el total de metros cuadrados que tienen las propiedades de “Mario Bros”.

Para optar a la máxima puntuación se debe hacer uso de estructuras, arrays y funciones, siendo éstas lo más genéricas posibles.

Ejercicio 1 [2 puntos]

Escribe un programa en C que contenga una función que pida dígitos de uno en uno al usuario. Dicha función debe construir un número entero con los dígitos leídos, tal y como se muestra en los ejemplos. La función debe dejar de solicitar dígitos cuando el número leído tenga más de un dígito. Dicha función debe devolver el número creado y el número de dígitos que tiene ese número. Finalmente, la función main, deberá invocar a la función creada e imprimir ambos valores. Con el objetivo de evaluar el paso de parámetros, no está permitido el uso de variables globales.

Ejemplo de entrada	Ejemplo de salida
1 2 3 4 5 10	5 54321
3 7 2 7 77	4 7273

Ejercicio 2 [3 puntos]

Escribe un programa en C que contenga una función que elimine los espacios en blanco de una cadena de caracteres pasada como parámetro. Además, la función debe devolver el número de espacios eliminado, así como el número de caracteres que sean dígitos (del 0 al 9).

El programa principal debe solicitar la cadena al usuario (máximo 50 caracteres), invocar a la función creada e imprimir la cadena modificada, el número de espacios detectado y el número de dígitos que contiene la cadena. Con el objetivo de evaluar el paso de parámetros, no está permitido el uso de variables globales. Aunque no es necesario su uso, se incluye la siguiente información sobre los caracteres ASCII, así como un ejemplo de salida (nótese que en el primer ejemplo de entrada hay tres espacios entre las palabras "Hola" y "mundo":

Carácter	Valor ASCII
‘ ’ (espacio)	32
‘0’	48
‘9’	57

Ejemplo de entrada	Ejemplo de salida
Hola mundo	Holamundo 3 0
Ca12 tao8 91 12 11	Ca12tao8911211 4 9

Ejercicio 3 [5 puntos]

Un importante banco quiere desarrollar una aplicación en C para controlar a sus clientes y las cuentas de los mismos. De cada cliente, es necesario almacenar, al menos, su nombre (hasta 50 caracteres), identificación (un entero) y localidad (hasta 20 caracteres). Cada cliente puede tener a su vez hasta 5 cuentas diferentes, de las cuales es necesario almacenar su número (un entero), el tipo de cuenta (puede ser corriente, de ahorro y nómina) y el saldo (número decimal).

Como el banco únicamente cuenta con los clientes más exclusivos, es suficiente con que la aplicación pueda almacenar 20 clientes diferentes.

Además, la aplicación debe incluir funciones que permitan:

1. Insertar un nuevo cliente, con su nombre, su identificación y población, pidiendo los datos necesarios al usuario. En caso de tener almacenados ya 20 clientes diferentes, esta función mostrará un error y no añadirá ningún cliente. Si ya se ha insertado algún cliente, la función no debe sobrescribirlo, almacenando el nuevo cliente sin modificar ni eliminar el anterior.
2. Insertar una nueva cuenta. Esta función debe pedir el identificador de un usuario y los datos necesarios para crear su cuenta actual. Si el usuario introducido ya tiene 5 cuentas, la función mostrará el error y no añadirá ninguna cuenta. Esta función no debe modificar ni eliminar las cuentas ya almacenadas de un cliente.
3. Modificar el saldo de una determinada cuenta de un determinado cliente. La función debe recibir como parámetros, al menos, el identificador del cliente, el número de su cuenta y un número decimal que indique el cambio en su cuenta (por ejemplo, 12 para añadir 12 euros o -23.50 para reducir su saldo en 23.50), y realizar la modificación solicitada. La función debe retornar información sobre si ha realizado la modificación o si el cliente indicado no tiene una cuenta con el identificador dado.
4. Mostrar el nombre e identificador de todos los clientes (sin repetir) que tengan alguna cuenta con un saldo negativo.
5. Mostrar el nombre e identificador del cliente que tenga más dinero total de una determinada localidad pasada como parámetro, siendo este la suma del saldo de todas sus cuentas.

Crea una pequeña función main que cree las variables necesarias para el correcto funcionamiento de la aplicación e invoque al menos una vez a cada una de las funciones anteriores con los parámetros necesarios.

Para optar a la máxima puntuación se debe hacer uso de estructuras, arrays y funciones, siendo éstas lo más genéricas posibles. Con el objetivo de evaluar el paso de parámetros, no está permitido el uso de variables globales.