

# PROGRAMACIÓN I

<b>Comenzado el</b>	domingo, 16 de diciembre de 2018, 22:08
<b>Estado</b>	Finalizado
<b>Finalizado en</b>	domingo, 16 de diciembre de 2018, 22:15
<b>Tiempo empleado</b>	7 minutos 5 segundos
<b>Puntos</b>	15,50/20,00
<b>Calificación</b>	<b>7,75</b> de 10,00 ( <b>78%</b> )



Correcta

Puntúa 1,00  
sobre 1,00

¿Cuál de las instrucciones comentadas con el símbolo // es INCORRECTA?

```
#include <stdio.h>
#include <string.h>

typedef struct
{
    char  nombre[30];
    int   codigo;
} Provincia;

void escribir (Provincia p)
{
    printf ("%s (%d)\n", p.nombre, p.codigo); // 1
}
```

```
int par (Provincia p){
    if (p.codigo%2) // 2
        return 1;
    return 0;
}

int main ()
{
    Provincia prov1, prov2;

    strcpy (prov1.nombre, "Madrid");
    prov1.codigo=91;

    printf ("%d", par(&prov1)); // 3
}
```

```
return 0;
}
```

Seleccione una:

- ☐ 2
- ☒ 3 ✓
- ☐ 1

**EXPLICACIÓN:**

La instrucción 3 es incorrecta porque la función `par` espera como argumento una variable de tipo `Provincia` y se pasa la dirección de una variable de tipo `Provincia`.

La respuesta correcta es: 3



## Pregunta 2


Correcta

Puntúa 1,00  
sobre 1,00

¿Qué vale  $*p < *q$  dadas las siguientes declaraciones?

```
int a[] = {5,15,34,54,14,2,52,72};  
int *p = &a[1];  
int *q = &a[5];
```

Seleccione una:

- ☐ 1
- ☐ ninguna de las anteriores
- ☒ 0 

### EXPLICACIÓN:

Sustituyendo los valores de la tabla, la expresión  $*p < *q$  es  $15 < 2$ . Por lo tanto, como es falsa, vale 0.

La respuesta correcta es: 0

### Pregunta 3

Incorrecta

Puntúa -0,50  
sobre 1,00

¿Qué imprime por pantalla el siguiente código? Supón que el usuario introduce los siguientes datos: alumno pass 1 2 1234

```
#include <stdio.h>
```

```
#define DIM 100
```

```
typedef struct {  
    int dia; /* Dia de nacimiento */  
    int mes; /* Mes de nacimiento */  
    int anyo; /* Anyo de nacimiento */  
} Fecha;
```

```
typedef struct {  
    char nombre [DIM]; /* Nombre de la persona */  
    char contra [DIM]; /* Contraseña de la persona */  
    Fecha fecha; /* Fecha de nacimiento */  
} Usuario;
```

```
int main() {
```

```
    Usuario us;  
    Fecha f;
```

```
    printf("Introduzca su nombre: ");  
    scanf("%s", us.nombre);  
    printf("Introduzca su contrasena: ");  
    scanf("%s", us.contra);
```


```
    printf("Introduzca su dia de nacimiento: ");
```

```
    scanf("%d", &f.dia);  
    printf("Introduzca su mes de nacimiento: ");  
    scanf("%d", &f.mes);  
    printf("Introduzca su anyo de nacimiento: ");  
    scanf("%d", &f.anyo);  
    printf("Nombre: %s\n", us.nombre);
```

```
    printf("Contrasena: %s\n", us.contra);  
    printf("Fecha de nacimiento: %d/%d/%d", us.fecha.dia, us.fecha.mes, us.fecha.anyo);
```

```
    return 0;  
}
```

Seleccione una:

- ☐ El nombre, la contraseña y valores indeterminados para la fecha. Cada dato en un línea
- ☐ Nada puesto que el programa no compila
- ☒ El nombre, la contraseña y la fecha (en formato dia/mes/año), cada dato en una línea 

### SOLUCIÓN:

Si estudiamos el código, el nombre y la contraseña del usuario se guardan en los miembros correspondientes de la variable `us`, mientras que los datos de la fecha (día, mes y año) se guardan dentro de otra variable de tipo `Fecha`. Al final, los datos que se imprimen son los almacenados en la variable `us`, que para el día, mes y año, son indeterminados

La respuesta correcta es: El nombre, la contraseña y valores indeterminados para la fecha. Cada dato en un línea

**Pregunta 4**

Incorrecta

Puntuación -0,50  
sobre 1,00

¿Cuál de las instrucciones comentadas con // (1, 2 o 3) es CORRECTA?

```
#include <stdio.h>
#include <string.h>

typedef struct
{
    char  nombre[30];
    int   codigo;
} Provincia;

void escribir (Provincia p)
{
    printf ("%s (%d)\n", p.nombre, p.codigo);
}
```

```
void copiar (Provincia p, Provincia * q){
    strcpy(q.nombre, p.nombre);    // 1
    *q->codigo = p.codigo;         // 2
}
```


```
int main ()
{
    Provincia prov1, prov2;

    strcpy (prov1.nombre, "Madrid");
    prov1.codigo=91;
    strcpy (prov2.nombre, "Burgos");
    prov2.codigo=947;

    copiar (prov1, &prov2);        // 3
```

```
    return 0;
}
```

Seleccione una:

- ☒ 2 
- ☐ 1
- ☐ 3

**EXPLICACIÓN:**

- La instrucción 1 es incorrecta porque `q.nombre` debería ser `q->nombre`.
- La instrucción 2 es incorrecta porque `*q->codigo` debería ser `(*q).codigo` o `q->codigo`.

La respuesta correcta es: 3

^

**Pregunta 5**

Correcta

Puntúa 1,00  
sobre 1,00

¿Cómo completarías la función volumenEsfera del siguiente programa para que calculase el volumen de una esfera de radio  $r$ ?

```
#include <stdio.h>
```

```
#define PI 3.141592
```

```
double volumenEsfera(double r) {
```

```
// Aquí iría el código que calcula el volumen de la esfera
```

```
}
```

```
int main() {
```

```
double radio = -1, volumen = 0;
```

```
do {
```

```
printf("Introduzca el radio de la esfera: ");
```

```
scanf("%lf", &radio);
```

```
if (radio <= 0)
```

```
printf("El radio debe ser mayor que 0\n");
```

```
}while (radio <= 0);
```

```
volumen = volumenEsfera(radio);
```

```
printf("El volumen de una esfera de radio %lf es %lf\n", radio, volumen);
```

```
return 0;
```

```
}
```

(NOTA: Recuerda que el volumen de una esfera es:  $V = \frac{4}{3} \pi r^3$ )

Seleccione una:



```
return 4*r^3*PI/3;
```



```
double vol;
```

```
vol=4.0*(r*r*r)*PI/3.0;  
return vol;
```



```
double vol;
```

```
vol=4*(r*3)*PI/3;  
return vol;
```

#### SOLUCIÓN:

La única forma posible de elevar al cubo de las tres formas propuestas es  $r*r*r$ .

La respuesta correcta es:

```
double vol;
```

```
vol=4.0*(r*r*r)*PI/3.0;  
return vol;
```

## Pregunta 6

Correcta

Puntúa 1,00  
sobre 1,00

En el siguiente código C, deseo reservar memoria para la variable especificada:

```
#include <stdio.h>
#include <string.h>
int main()
{
char cad [120];
char *ptr;
int lon;
printf ("\nIntroduce una linea de texto\n") ;
gets(cad) ;
lon = strlen(cad) ;
/* reservo memoria*/
strcpy (ptr, cad);
printf ("ptr = %s",ptr) ;
free(ptr);
return 0;
}
```

Seleccione una:



ptr = (char\*) malloc ((lon+1)\* sizeof (char));



ptr = (char ) malloc ((lon)\* sizeof (char));



ptr= (char ) malloc ((lon)\*sizeof(NULL));

### EXPLICACIÓN:

Se trata de reservar memoria para una variable de cadena y ptr es un puntero a una cadena. El tamaño de la memoria requerida queda determinado por el tipo de variable (char) y lo que ocupa en memoria multiplicado por el número total de caracteres de la mas un byte extra por el carácter `'/0'` Al final de línea. Ptr devuelve un puntero que apunta a una sección de memoria capaz de contener una cadena.

La respuesta correcta es:

ptr = (char\*) malloc ((lon+1)\* sizeof (char));





**Pregunta 7**

Correcta

Puntúa 1,00  
sobre 1,00

¿Qué muestra por pantalla el siguiente programa?


```
int main()
{
char primera='C', segunda='A';

primera -= segunda;

printf("%d", primera);
return 0;

}
```

Seleccione una:

- ☐ A
- ☐ C
- ☒ 2 

**SOLUCIÓN:**

La instrucción `primera -= segunda;` equivale a `primera = primera - segunda;`

Por lo tanto sería `primera = 'C' - 'A';`

Restando los valores ASCII se obtendría el valor 2.

La respuesta correcta es: 2

**Pregunta 8**

Incorrecta

Puntúa -0,50  
sobre 1,00

¿Qué escribe el siguiente programa?

#include &lt;stdio.h&gt;


```
void f (int * p)
{
    *p=1;
    printf ( "%d", *p);
}
```

```
int main()
{
    int x=2;
```

```
f (&x);
printf( "%d", x);
```

```
return 0;
}
```

Seleccione una:

- ☒ 12 
- ☐ Ninguna de las otras
- ☐ 11

**EXPLICACIÓN:**

El primer printf que se ejecuta es el de la función f y escribe \*p, que vale 1. El segundo printf que se ejecuta es el de main, que escribe el valor de la x de main.

La llamada f(&x) hace que p valga &x. Por lo tanto, \*p=1 es lo mismo que x=1. Este es el valor que se escribe desde main.

La respuesta correcta es: 11

**Pregunta 9**

Correcta

Puntúa 1,00  
sobre 1,00

¿Con qué formato deberíamos llamar a la función fopen para que cada vez que se abra el fichero y se escriba en él, se añada una nueva línea con el texto "hola", como en el ejemplo siguiente?

```
int main() {
```

```
FILE *f;
```

```
int i;
```

```
for (i=0;i<10;i++)
```

```
{
```

```
    f=fopen("fichero.txt", OPCION);
```

```
    fprintf(f, "hola\n");
```

```
    fclose(f);
```

```
}
```

```
return 0;
```

```
}
```

Seleccione una:

- ☒ "a" ✓
- ☐ "w+"
- ☐ "w"

**EXPLICACIÓN:**

El formato "a" abre el fichero de forma que la información que se escriba se añada a la ya existente, mientras que los formatos con "w" empiezan a escribir desde el principio del fichero, borrando todo lo que contuviese anteriormente.

La respuesta correcta es: "a"



### Pregunta 10

Correcta


Puntúa 1,00  
sobre 1,00

Dado el siguiente programa

```
int main()
{
    int a=1;
    for (;!a;) {
        a++;
        if (a==10)
            break;
    }
    printf("%d\n",a);
    return 0;
}
```

¿Qué se escribe por pantalla?

Seleccione una:

- ☐ 10
- ☐ 11
- ☒ 1 

EXPLICACIÓN:

El programa no entra en el bucle for ya que !a es equivalente a la comprobación a==0

La respuesta correcta es: 1

### Pregunta 11

Correcta

Puntúa 1,00  
sobre 1,00

Si tenemos definidos:

```
#define DIM 30  
#define U 4
```

```
typedef struct {
```

```
    int dia;  
    int mes;  
    int anyo;
```

```
} Fecha;
```

```
typedef struct {
```

```
    char nombre [DIM];  
    char contra [DIM];  
    Fecha fecha;
```

```
} Usuario;
```

```
typedef struct {
```

```
    int numUsuarios;  
    Usuario usuario [U];
```

```
} Sesion;
```

Y queremos que el siguiente bucle finalice cuando encuentre un usuario registrado con el mismo nombre que el usuario introducido, ¿dónde debemos colocar la instrucción *break*?

```
#include <stdio.h>  
#include <string.h>
```

```
int main() {
```

```
Usuario usuario;  
Sesion sesion;  
int i;
```

```
sesion.numUsuarios = 0;  
printf("Introduzca nombre: ");  
scanf("%s", usuario.nombre);  
printf("Introduzca contrasena: ");  
scanf("%s", usuario.contra);  
printf("Introduzca dia de nacimiento: ");  
scanf("%d", &usuario.fecha.dia);  
printf("Introduzca mes de nacimiento: ");  
scanf("%d", &usuario.fecha.mes);  
printf("Introduzca anyo de nacimiento: ");  
scanf("%d", &usuario.fecha.anyo);
```

```
for (i = 0; i < sesion.numUsuarios; i++)  
{
```

```
    //(1)  
    if (strcmp(sesion.usuario[i].nombre, usuario.nombre) == 0) {  
        //(2)  
    }
```

```
    //(3)
```

```
}
```

```
if (i==sesion.numUsuarios) {  
    printf("El usuario %s se ha registrado correctamente.\n", usuario.nombre);  
    sesion.usuario[sesion.numUsuarios] = usuario;  
    sesion.numUsuarios++;  
}
```

```
else
```

```
    printf("El usuario %s ya se encuentra en la sesion.\n", usuario.nombre);
```

```
return 0;
```

```
}
```

Selecione una:

☐ (3)

- ☒ (2) ✓
- ☐ (1)

La respuesta correcta es: (2)

### Pregunta 12

Correcta

Puntúa 1,00  
sobre 1,00

Si al ejecutar el siguiente trozo de código

```
char nombre_ciudad[20];  
printf("Introduzca un nombre de ciudad: ");  
scanf("%s", nombre_ciudad);
```

el usuario introduce la cadena Madrid y pulsa INTRO, ¿cuántos elementos de la cadena se ocuparán?

Seleccione una:

- ☒ 7 ✓
- ☐ 6
- ☐ 5

SOLUCIÓN:

Se ocuparán 7 elementos: 6 caracteres de la cadena Madrid y uno correspondiente al código ASCII 0 de fin de cadena.

La respuesta correcta es: 7

### Pregunta 13

Correcta

Puntúa 1,00  
sobre 1,00

¿Cuántas veces se mostrará el menú de inicio antes de terminar el programa si el usuario va a teclear como opción 1?

```
#include <stdio.h>
```

```
typedef enum {altaUsuario=1, nuevaSesion, salir} MenuInicial;
```

```
int main() {
```

```
    int opcion=salir;
```



```
while (opcion!=salir)
```

```
{
```

```
printf ("\n");
```

```
printf ("1. Dar de alta a una nueva persona.\n");
```

```
printf ("2. Iniciar una sesion para un usuario.\n");
```

```
printf ("3. Salir del programa.\n\n");
```

```
printf ("Elija una opcion: ");
```

```
scanf ("%d", &opcion);
```

```
switch (opcion)
```

```
{
```

```
case altaUsuario:
```

```
printf("\nAlta usuario\n\n");
```

```
break;
```

```
case nuevaSesion:
```

```
printf("\nNueva sesion.\n\n");
```

```
break;
```

```
case salir:
```

```
printf ("\nAdios.\n\n");
```

```
break;
```

```
default:
```

```
printf ("\nOpcion incorrecta.\n\n");
```

```
}
```

```
}
```

```
return 0;
```

```
}
```

Seleccione una:

- ☐ Hasta que el usuario introduzca '3'.
- ☒ Ninguna. ✓
- ☐ Una.

SOLUCIÓN:

La variable `opcion` se inicializa al valor `salir`, y la condición de mantenimiento del bucle es que `opcion != salir`. No se llega a ejecutar una sólo vez ya que en la primera iteración no se cumple la condición.

La respuesta correcta es: Ninguna.

### Pregunta 14

Correcta

Puntúa 1,00  
sobre 1,00

¿Qué escribe el siguiente programa?

```
#include <stdio.h>
```

```
void fun (int * a) {  
    int i;  
    *a = 0;  
}  
  
int main ()  
{  
    int i;  
    int x[] = {1,1,1};  
  
    fun (&x[2]);  
  
    for (i=0; i<3; i++)  
        printf ("%d ", x[i]);  
}
```

```
    return 0;  
}
```

Seleccione una:

- ☐ Ninguna de las otras
- ☐ 1 1 1
- ☒ 1 1 0 ✓

#### EXPLICACIÓN:

La llamada `fun(&x[2]);` hace que `a` dentro de `fun` apunte al tercer elemento del array `x` de `main` (los índices de los arrays de C empiezan por 0). Por eso, `*a=0;` hace que el tercer elemento del array `x` de `main` se ponga a 0.

La respuesta correcta es: 1 1 0

### Pregunta 15

Correcta

Puntúa 1,00  
sobre 1,00

Analiza el siguiente programa:

```
#include <stdio.h>
```

```
#define SIZE 100
```

```
int main(){
```

```
    char nombre[SIZE];
```

```
    printf("Introduce tu nombre: ");
```

```
    scanf("%s", nombre);
```

```
    printf("%s", nombre);
```

```
    return 0;
```

```
}
```

Si durante la ejecución, el usuario escribe el nombre "*Bruno Perez*" cuando se lo indican por pantalla, ¿qué salida emitirá el segundo *printf*?

Seleccione una:

- ☐ Bruno Perez
- ☐ Dará un error en tiempo de ejecución
- ☒ Bruno ✓

Respuesta correcta

La función *scanf* no recoge los espacios, por lo que acabará de leer la cadena en el espacio entre *Bruno* y *Perez*, almacenando únicamente el valor "Bruno" en la variable.

La respuesta correcta es: Bruno

**Pregunta 16**

Correcta

Puntúa 1,00  
sobre 1,00

Utilizando la siguiente enumeración, ¿cómo podemos comprobar que el mes introducido por el usuario es válido?

```
typedef enum {enero=1, febrero, marzo, abril, mayo, junio, julio,  
             agosto, septiembre, octubre, noviembre, diciembre} Mes;
```

```
Mes mes;
```

```
scanf("%d", &mes);
```

Seleccione una:



```
if (mes==enero, febrero, marzo, abril, mayo, junio, julio,  
    agosto, septiembre, octubre, diciembre)  
    printf("Mes valido\n");  
else  
    printf("Mes invalido\n");
```



```
if (mes==enero && mes==febrero && mes==marzo && mes==abril && mes==mayo &  
& mes==junio  
    && mes==julio && mes==agosto && mes==septiembre && mes==octubre && me  
s==diciembre)  
    printf("Mes valido\n");  
else  
    printf("Mes invalido\n");
```



```
if (mes>=enero && mes<=diciembre)  
    printf("Mes valido\n");  
else  
    printf("Mes invalido\n");
```

**SOLUCIÓN:**

```
if (mes==enero, febrero, marzo, abril, mayo, junio, julio, agosto, septiemb  
r, octubre, diciembre)  
    printf("Mes valido\n");  
else  
    printf("Mes invalido\n");
```

NO es una sintaxis válida en C



```
if (mes==enero && mes==febrero && mes==marzo && mes==abril && mes==mayo && mes==junio && mes==julio && mes==agosto && mes==septiembre && mes==octubre && mes==diciembre)
    printf("Mes valido\n");
else
    printf("Mes invalido\n");
```

También es incorrecta dado que la expresión indica que un mes valido es aquel coincide con todos los meses del año.

La respuesta correcta es:

```
if (mes>=enero && mes<=diciembre)
    printf("Mes valido\n");
else
    printf("Mes invalido\n");
```

**Pregunta 17**

Correcta

Puntúa 1,00  
sobre 1,00

El problema del siguiente proyecto se pone de manifiesto durante ...

1. El archivo receta.h:

```
double receta (int x, int * y);
```

2. El archivo receta.c:

```
#include "receta.h"
double receta (int x, int y) {
    return (x+y)/2.0;
}
```

3. El archivo main.c:

```
#include <stdio.h>
#include "receta.h"
int main () {
    double resultado = receta (1,2);
    printf (".2lf\n", resultado);
    return 0;
}
```

Seleccione una:

- ☐ el enlazado del proyecto
- ☒ la compilación de receta.c ✓
- ☐ la compilación de main.c

**EXPLICACIÓN:**

El prototipo de la función receta en el archivo receta.h

```
double receta (int x, int * y);
```

no coincide con la definición del archivo receta.c (el tipo del segundo parámetro es int en la declaración e int \* en la definición):

```
double receta (int x, int y) {
    return (x+y)/2.0;
}
```

Esto da lugar a un error al compilar el archivo receta.c.

La respuesta correcta es: la compilación de receta.c



**Pregunta 18**

Correcta

Puntúa 1,00  
sobre 1,00

Teniendo un programa en el que queremos inicializar la variable matrix con ceros en todas sus posiciones, ¿qué fragmento de código realiza la inicialización correctamente?:

```
int main() {  
  
    int matrix[3][3];  
    int i, j;  
  
    // Código para inicializar la matriz  
  
    return 0;  
}
```

Seleccione una:



```
for(i = 0; i < 3; i++){  
    for(j = 0; j < 3; j++){  
        matrix[i][j] = 0;  
    }  
}
```



```
for(i = 0; i < 3; i++){  
    matrix[i][0] = 0;  
}  
  
for(j = 0; j < 3; j++){  
    matrix[0][j] = 0;  
}
```



```
for(i = 0; i < 3; i++){  
    for(j = 0; j < i; j++){  
        matrix[i][j] = 0;  
    }  
}
```

Respuesta correcta



```
/* Este código sólo inicializa la primera fila y la primera columna de la matriz */
```

```
for(i = 0; i < 3; i++){  
    matrix[i][0] = 0;  
}
```

```
for(j = 0; j < 3; j++){  
    matrix[0][j] = 0;  
}
```

```
/* Este código sólo inicializa algunos elementos de la matriz */
```

```
for(i = 0; i < 3; i++){  
    for(j = 0; j < i; j++){  
        matrix[i][j] = 0;  
    }  
}
```

```
/* Este código sí recorre todas las posiciones de la matriz e inicializa a cero todos los elementos */
```

```
for(i = 0; i < 3; i++){  
    for(j = 0; j < 3; j++){  
        matrix[i][j] = 0;  
    }  
}
```

La respuesta correcta es:

```
for(i = 0; i < 3; i++){  
    for(j = 0; j < 3; j++){  
        matrix[i][j] = 0;  
    }  
}
```

### Pregunta 19

Correcta

Puntúa 1,00  
sobre 1,00

¿Para qué sirve la instrucción break cuando aparece en un caso de la instrucción switch?

Seleccione una:

- ☐ para saltar las instrucciones que aparezcan hasta la siguiente llave de cierre }
- ☐ para terminar la ejecución del programa
- ☒ para salir de la instrucción switch ✓

^

### SOLUCIÓN:

La instrucción break cuando aparece en un caso de la instrucción switch hace que salgamos de la instrucción switch y no examinemos el resto de los casos.

La respuesta correcta es: para salir de la instrucción switch

## Pregunta 20

Correcta

Puntúa 1,00  
sobre 1,00

¿Qué escribe el siguiente programa?

```
#include <stdio.h>
void g (int x)
{
    x=0;
}
void f (int * p)
{
    *p=1;
    g (*p);
}
int main() {
    int x=2;

    f (&x);

    printf("%d", x);

    return 0;
}
```

Seleccione una:

- ☐ 0
- ☐ 2
- ☒ 1 

### EXPLICACIÓN:

La función f altera el valor de la variable x de main, que pasa a valer 1 en la instrucción \*p=1. Sin embargo, la función g solo cambia el valor de su variable local x, no de la variable local de f.

La respuesta correcta es: 1

Volver a: General ➞