

# PROGRAMACIÓN I

**Comenzado el** sábado, 15 de diciembre de 2018, 13:35

**Estado** Finalizado

**Finalizado en** sábado, 15 de diciembre de 2018, 13:56

**Tiempo empleado** 21 minutos 12 segundos

**Puntos** 18,50/20,00

**Calificación** 9,25 de 10,00 (93%)

## Pregunta 1

Correcta

Puntúa 1,00  
sobre 1,00

El formato “%4d”:

Seleccione una:

- ☒ Escribe un entero a la derecha de un campo de anchura 4 ✓
- ☐ Escribe un entero a la izquierda de un campo de anchura 4
- ☐ Escribe un entero utilizando 4 dígitos

### SOLUCIÓN:

%4d escribe un entero a la derecha de un campo de anchura 4, tal y como se especifica y solicita en el enunciado del ejercicio 3 de la práctica 2.

La respuesta correcta es: Escribe un entero a la derecha de un campo de anchura 4

Correcta

Puntúa 1,00  
sobre 1,00

¿Qué hace el siguiente programa en C?

#include &lt;stdio.h&gt;

#define NUM 50

```
int main() {  
  
    int i = 0;  
  
    while ( i < NUM){  
        printf("%d\n", i);  
        i += 2;  
    }  
  
    return 0;  
}
```

Seleccione una:

- ☐ Escribe todos los números pares hasta 50, inclusive
- ☐ Escribe todos los números del 0 al 50
- ☒ Escribe todos los números pares menores de 50 ✓

Respuesta correcta

Al inicial el bucle con  $i = 0$  e imprimir su valor, aumentándolo de dos en dos mientras que sea menor que NUM, que es una macro que representa el valor 50, obtendremos una lista con todos los números pares menores que 50.

La respuesta correcta es: Escribe todos los números pares menores de 50

Correcta

Puntúa 1,00  
sobre 1,00

Dado el siguiente programa, ¿qué se mostraría por pantalla?

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

```
#define X 100  
#define Y 50  
#define Z 10  
#define TOTAL X + (Y + Z)
```

```
int main() {  
    printf("%d\n", TOTAL*2);  
    return 0;  
}
```

Seleccione una:

- ☐ 170
- ☐ 320
- ☒ 220 ✓

Respuesta correcta

SOLUCIÓN:

El compilador sustituye las macros literalmente, por lo que la operación que se imprime es  $100 + (50 + 10) * 2$ , cuyo resultado, aplicando la prioridad de operadores, es 220.

La respuesta correcta es: 220

**Pregunta 4**

Correcta

Puntúa 1,00  
sobre 1,00

1. Para saber si dos cadenas son iguales usamos strcmp. ¿Cómo funciona?

Seleccione una:

- ☒ strcmp recibe como argumento las dos cadenas y devuelve 0 si son iguales y otro valor si son diferentes ✓
- ☐ strcmp recibe como argumento las dos cadenas y devuelve *true* si son iguales y *false* si son diferentes
- ☐ strcmp recibe como argumento las dos cadenas y devuelve la cadena mayor

SOLUCIÓN:

El funcionamiento de strcmp se especifica y prueba en el ejercicio 5 del tema 3.3.

La respuesta correcta es: strcmp recibe como argumento las dos cadenas y devuelve 0 si son iguales y otro valor si son diferentes

^

Correcta

Puntúa 1,00  
sobre 1,00

Dada la siguiente función

```
int par(int numero)
```

```
{
```

```
    if((numero%2)==0)
```

```
        return 1;
```

```
    else
```

```
        return 0;
```

```
}
```

¿Cuál de los siguientes trozos de código sería correcto?

Seleccione una:

- ☐

```
int main()
{
    int res;
    res = par();
```
- ☒

```
int main()
{
    int nu, res;
    res = par(nu);
```

 ✓
- ☐

```
int main()
{
    int nu, res;
    par(nu, res);
```

EXPLICACIÓN:

Se trata de una función que recibe un argumento y devuelve un valor.

La respuesta correcta es: 

```
int main()
```

```
{
    int nu, res;
    res = par(nu);
```

Correcta

Puntúa 1,00  
sobre 1,00

¿Qué escribe el siguiente programa?

```
#include <stdio.h>
void f (int x)
{
    x=1;
    printf ("%d", x);
}
int main() {
    int x=2;

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

    return 0;
}
```

Seleccione una:

- ☐ 11
- ☐ 22
- ☒ 12 ✓

**EXPLICACIÓN:**

El primer printf que se ejecuta es el que se encuentra dentro de f y escribe 1. El siguiente printf, el de main, escribe el valor de x de main, que es 2 porque f no cambia el valor de la variable x de main.

La respuesta correcta es: 12

Correcta

Puntúa 1,00  
sobre 1,00

¿Qué escribe el siguiente programa?

```
#include <stdio.h>
int main ()
{
    int n;
    char cadena [30];
    FILE * f;

    f = fopen ("datos.txt", "r");
    if (!f)
        return 1;

    for (n=0; fscanf (f, "%s", cadena) == 1; n++);
    printf ("%d\n", n);

    return 0;
}
```

si el archivo datos.txt contiene el siguiente texto:

Se trata de un acuerdo aceptado por todas

Seleccione una:

- ☐ 9
- ☒ 8 ✓
- ☐ 10

**EXPLICACIÓN:**

Recuerda que fscanf devuelve un entero igual al número de formatos de la instrucción, siempre que la lectura haya tenido éxito.

Por lo tanto, el bucle

```
for (n=0; fscanf (f, "%s", cadena) == 1; n++);
```

hace que la variable n se incremente cada vez que se lee una cadena de caracteres del archivo. El número de cadenas es 8.

La respuesta correcta es: 8

Correcta

Puntúa 1,00  
sobre 1,00

Si en nuestro programa principal tenemos definido

```
#define DIM 20
```

```
char nombre[DIM];
```

```
Sesion s;
```

```
int retorno;
```

y queremos llamar a la función con prototipo:

```
int buscarUsuarioPorNombre(Sesion sesion, char * nombre);
```

¿Cual de las siguientes llamadas es correcta?

Seleccione una:

- ☐ retorno = buscarUsuarioPorNombre (s, &nombre);
- ☐ retorno = buscarUsuarioPorNombre (s, nombre[DIM]);
- ☒ retorno = buscarUsuarioPorNombre (s, nombre); ✓

**EXPLICACIÓN:**

La llamada se debe hacer como:

```
retorno = buscarUsuarioPorNombre (s, nombre);
```

porque la variable nombre ya es de tipo char\* al estar declarada como una cadena de caracteres.

La respuesta correcta es: retorno = buscarUsuarioPorNombre (s, nombre);

**Pregunta 9**

Incorrecta

Puntúa -0,50  
sobre 1,00

El siguiente programa escrito en C no funciona bien:

```
#define DIM 64
```

```
typedef struct {  
    int dia;  
    int mes;  
    int anyo;  
} Fecha;
```

```
typedef struct {  
    char nombre [DIM];  
    char apellidos [DIM];  
    Fecha* fecha_nacimiento;  
} Contacto;
```

```
void iniContacto(Contacto* cnt, char* nmb, char* apII, int dn, int m  
n, int an){  
    Fecha fn;
```

```
    strcpy(cnt->nombre, nmb);  
    strcpy(cnt->apellidos, apII);  
  
    cnt->fecha_nacimiento= &fn;  
  
    cnt->fecha_nacimiento->dia= dn;  
    cnt->fecha_nacimiento->mes= mn;  
    cnt->fecha_nacimiento->anyo= an;  
    return;  
}
```

```
void impContacto(Contacto* cnt){  
    printf("%s %s (%d-%d-%d)\n", cnt->nombre, cnt->apellidos,  
    cnt->fecha_nacimiento->dia, cnt->fecha_nacimiento->mes,  
    cnt->fecha_nacimiento->anyo);  
    return;  
}
```

```
int main(int argc, char** argv) {
```

^



```
Contacto amigo1, amigo2;
```

```
iniContacto(&amigo1, "Manuel","Molina",3,3,1999);  
iniContacto(&amigo2, "Antonio","Moro",12,1,1997);
```

```
impContacto(&amigo1);  
impContacto(&amigo2);
```

```
return 0;
```

```
}
```

Al ejecutarlo aparece por pantalla:

Manuel Molina (12-1-1997)  
Antonio Moro (4202528-2673824-2673888)

¿Cuál de las siguientes afirmaciones es correcta?

Seleccione una:

- ☒ Dentro de la función *iniContacto()* deberían sustituirse todos los *"fecha\_nacimiento->"* por *"fecha\_nacimiento."*. ❌
- ☐ La variable *fn* es local a la función *iniContacto()*, por lo que no es adecuado utilizarla para inicializar el campo *fecha\_nacimiento* del contacto apuntado por el argumento *cnt*.
- ☐ Dentro de la función *iniContacto()* deberían sustituirse la asignación *"cnt->fecha\_nacimiento= &fn"* por *"cnt->fecha\_nacimiento= fn"*.

Respuesta incorrecta.

El problema es que se utiliza la dirección de la variable *fn* declarada dentro de la función *iniContacto()* para inicializar el campo *fecha\_nacimiento* y dicha variable deja de existir al terminar la ejecución de la función, por lo que el espacio de memoria que ocupaba puede reutilizarse para otra cosa.

La respuesta correcta es: La variable *fn* es local a la función *iniContacto()*, por lo que no es adecuado utilizarla para inicializar el campo *fecha\_nacimiento* del contacto apuntado por el argumento *cnt*.

### Pregunta 10

Correcta

Puntúa 1,00  
sobre 1,00

¿Qué instrucción del siguiente programa habría que modificar para que funcione correctamente?


```
#define NUM 20
```

```
int main()
{
    int i, * t; // 1

    t= (int *) malloc(NUM); // 2
    for(i=0;i<NUM; i++)
        t[i]=0;

    return 0;
}
```

Seleccione una:

- ☒ La instrucción 2 
- ☐ Ninguna, el programa es correcto
- ☐ La instrucción 1

La instrucción correcta sería:

```
t = (int *) malloc(NUM*sizeof(int));
```

La respuesta correcta es: La instrucción 2

**Pregunta 11**

Correcta

Puntúa 1,00  
sobre 1,00

¿Qué escribe el siguiente programa si se teclea 7?

```
#include <stdio.h>

int main () {
    int x;

    scanf ("%d", &x);

    if (x<3 || x>=1)
        printf ("1");
    else if (x<8)
        printf ("2");
    else
        printf ("3");

    return 0;
}
```

Seleccione una:

- ☒ 1 ✓
- ☐ 2
- ☐ 3

**SOLUCIÓN:**

Se cumple la primera condición pues x es mayor o igual que 1, por lo que muestra un 1.

La respuesta correcta es: 1

## Pregunta 12

Correcta

Puntúa 1,00  
sobre 1,00

Si en una función main aparecen las siguientes instrucciones

```
int *pnumero;
```

```
int num1, num2;
```

```
char *pchar;
```

```
char letra1;
```

```
num1 = 2;
```

```
num2 = 5;
```

```
letra1 = 'a';
```

```
pchar = &letra1;
```

```
*pchar = 'B';
```

¿Cómo obtengo el valor de **letra1**?

Seleccione una:

- ☐ Con &letra1 o pchar
- ☐ Con \*letra1 o &pchar
- ☒ Con letra1 o \*pchar ✓

EXPLICACIÓN:

Como pchar apunta letra1, un equivalente a letra1 es \*pchar

La respuesta correcta es: Con letra1 o \*pchar

### Pregunta 13

Correcta

Puntúa 1,00  
sobre 1,00

¿Qué escribe el siguiente programa?

```
#include <stdio.h>

int main() {
    int i, j;

    for (i=0;i<2;i++)
        for(j=0;j<3;j++)
            printf("%d", i);

    return 0;
}
```

Seleccione una:

- ☐ 012012
- ☐ 010101
- ☒ 000111 ✓

#### SOLUCIÓN:

- La variable i se inicializa a 0.
- Como  $i < 2$  ( $0 < 2$ ) es cierto, se ejecuta el bucle interior:
  - La variable j se inicializa a 0.
  - Como  $j < 3$  ( $0 < 3$ ) es cierto, se ejecuta el cuerpo del bucle interior:
    - Se escribe i, es decir, 0.
    - Se incrementa j, que pasa a valer 1.
  - Como  $j < 3$  ( $1 < 3$ ) es cierto, se ejecuta el cuerpo del bucle interior:
    - Se escribe i, es decir, 0.
    - Se incrementa j, que pasa a valer 2.
  - Como  $j < 3$  ( $2 < 3$ ) es cierto, se ejecuta el cuerpo del bucle interior:
    - Se escribe i, es decir, 0.
    - Se incrementa j, que pasa a valer 3.
  - Como  $j < 3$  es falso, se acaba el bucle interior.
  - Se incrementa i, que pasa a valer 1.
- Como  $i < 2$  ( $1 < 2$ ) es cierto, se ejecuta el bucle interior:
  - La variable j se inicializa a 0.
  - Como  $j < 3$  ( $0 < 3$ ) es cierto, se ejecuta el cuerpo del bucle interior:
    - Se escribe i, es decir, 1.
    - Se incrementa j, que pasa a valer 1.
  - Como  $j < 3$  ( $1 < 3$ ) es cierto, se ejecuta el cuerpo del bucle interior:
    - Se escribe i, es decir, 1.
    - Se incrementa j, que pasa a valer 2.
  - Como  $j < 3$  ( $2 < 3$ ) es cierto, se ejecuta el cuerpo del bucle interior:
    - Se escribe i, es decir, 1.
    - Se incrementa j, que pasa a valer 3.
  - Como  $j < 3$  es falso, se acaba el bucle interior.
  - Se incrementa i, que pasa a valer 2.
- Como  $i < 2$  ( $2 < 2$ ) es falso, se acaba el bucle exterior.

En resumen, el programa escribe 000111.

La respuesta correcta es: 000111

### Pregunta 14

Correcta

Puntúa 1,00  
sobre 1,00

Dado el siguiente programa:

```
#include <stdio.h>

int main() {
    int i,num;
    double fa;
    printf("\nCalculo el factorial de: ");
    scanf("%d",&num);
    fa=1;
    // Código para calcular el factorial
    /* solucion */
    printf("\nEl factorial de %d es %lf\n",num,fa);
    return 0;
}
```

¿Cuál de las siguientes es la solución para el cálculo del factorial?

Seleccione una:

- ☐ fa=fa\*i;
- ☐ for(i=1;i<=num;i++) fa\*\*;
- ☒ for(i=1;i<=num;i++) fa=fa\*i; ✓

#### SOLUCIÓN:

De acuerdo con la ecuación que define el factorial se puede realizar con un bucle for, y en cada iteración guardar el valor anterior que tiene volver a ser multiplicado.

La respuesta correcta es: for(i=1;i<=num;i++) fa=fa\*i;

### Pregunta 15

Correcta

Puntúa 1,00  
sobre 1,00

Dado el siguiente trozo de código

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

¿qué instrucción deberíamos añadir para leer nombres de ciudades de más de una palabra (por ejemplo "Ciudad Real")?

Seleccione una:

- ☐ scanf("%s", nombre\_ciudad);
- ☒ gets(nombre\_ciudad); ✓
- ☐ scanf("%c", nombre\_ciudad);

#### SOLUCIÓN:

La función gets nos permite leer por teclado una cadena de caracteres que tenga espacios.

La respuesta correcta es: gets(nombre\_ciudad);

^

**Pregunta 16**

Correcta

Puntúa 1,00  
sobre 1,00

¿Qué muestra por pantalla el siguiente programa?

```
#include <stdio.h>

int main() {
    int i, sum;
    for (i = 0, sum = 5; i < 5; i += 2)

        sum += i;
    printf("%d", sum);
    return 0;
}
```

Seleccione una:

- ☐ 13
- ☐ 7
- ☒ 11 ✓

SOLUCIÓN: Muestra el valor 11 ya que la condición de parada es  $i < 5$ 

La respuesta correcta es: 11

**Pregunta 17**

Correcta

Puntúa 1,00  
sobre 1,00

La definición (declaración) de variables en un programa "C":

Seleccione una:

- ☐ Las variables se pueden declarar o no, dependiendo del programador
- ☒ Si no declaramos una variable antes de usarla, el programa no funcionará ✓
- ☐ No es obligatoria, se pueden utilizar las variables sin declararlas

SOLUCIÓN:

Las variables sólo se pueden utilizar una vez declaradas. La declaración de variables es obligatoria para poder usarse.

La respuesta correcta es: Si no declaramos una variable antes de usarla, el programa no funcionará

**Pregunta 18**

Correcta

Puntúa 1,00  
sobre 1,00

Si en un proyecto grande cambiamos algo en un solo .c, ¿qué tenemos que hacer para ejecutarlo?

Seleccione una:

- ☐ Recompilar todos los .c y enlazar sus .o.
- ☒ Recompilar sólo el fichero .c modificado y enlazar de nuevo todos los .o. ✓
- ☐ Sólo recompilar el fichero .c modificado.

**EXPLICACIÓN:**

Es preciso recompilar únicamente el fichero que haya cambiado pero a continuación se debe enlazar de nuevo todos los .o para que el nuevo ejecutable refleje los cambios en el código  
La respuesta correcta es: Recompilar sólo el fichero .c modificado y enlazar de nuevo todos los .o.

**Pregunta 19**

Correcta

Puntúa 1,00  
sobre 1,00

¿Qué muestra por pantalla el siguiente programa?

```
#include <stdio.h>
void f1(int x, int *y, int a, int b)
{
    x=x+1;
    *y=*y+1;
    x=x+a;
    *y=*y+b;
    printf(" %d %d ",x,*y);
}
int main()
{
    int a=0,b=0;
    f1(a,&b,a,b);
    printf(" %d %d\n",a,b);
    return 0;
}
```

Seleccione una:

- ☐ 0 1 0 1
- ☐ 0 0 1 0
- ☒ 1 1 0 1 ✓

**EXPLICACIÓN:**

Dentro de la función x toma el valor de a y \*y el valor de b, por lo tanto dentro de la función el printf imprime 1 1 por pantalla.  
Al salir de la función sólo se mantiene la modificación de b (que es la única variable que se ha pasado por referencia) por lo que el printf imprime 0 1  
La respuesta correcta es: 1 1 0 1



## Pregunta 20

Correcta

Puntúa 1,00  
sobre 1,00

¿Qué escribe el siguiente programa?

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

```
typedef enum {verde=1, rojo, amarillo} Color;  
int f (int a) {  
    int i;  
    for (i=2; i<a-1; i++)  
        if (a%i==0)  
            return 1;  
    return 0;  
}  
int main () {  
    printf ("%d", f(7));  
    return 0;  
}
```

Seleccione una:

- ☐ 2
- ☒ 0 ✓
- ☐ 1

### EXPLICACIÓN:

La función f devuelve 0 si su argumento es primo. Como 7 es primo, f(7) es 0.

La respuesta correcta es: 0

Volver a: General ➡