

PROGRAMACIÓN I

Comenzado el	sábado, 15 de diciembre de 2018, 22:48
Estado	Finalizado
Finalizado en	sábado, 15 de diciembre de 2018, 22:57
Tiempo empleado	9 minutos 17 segundos
Puntos	18,50/20,00
Calificación	9,25 de 10,00 (93%)



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;

```

```

}

```

Seleccione una:

☐ (1)

☐ (3)

☒ (2) ✓

La respuesta correcta es: (2)

Si solicitamos dos fechas de la siguiente manera:

```
typedef struct{
```

```
    int dia;
```

```
    int mes;
```

```
    int anyo;
```

```
} Fecha;
```

```
Fecha fecha1, fecha2;
```

```
printf("Por favor, introduzca la primera fecha: ");
```

```
scanf("%d %d %d", &fecha1.dia,&fecha1.mes, &fecha1.anyo);
```

```
printf("Por favor, introduzca la segunda fecha: ");
```

```
scanf("%d %d %d", &fecha2.dia, &fecha2.mes, &fecha2.anyo);
```

¿Cuál de las siguientes opciones pinta por pantalla la fecha más antigua?

Seleccione una:



```
if (fecha1 < fecha2)
    printf("%d %d %d\n", fecha1);
else
    printf("%d %d %d\n", fecha2);
```



```
if ((fecha1.anyo < fecha2.anyo) &&
    (fecha1.mes < fecha2.mes) &&
    (fecha1.dia < fecha2.anyo))
    printf("%d %d %d\n", fecha1.dia, fecha1.mes, fecha1.anyo);
else
    printf("%d %d %d\n", fecha2.dia, fecha2.mes, fecha2.anyo);
```



```
if ((fecha1.anyo < fecha2.anyo) ||
    ((fecha1.anyo == fecha2.anyo) && (fecha1.mes < fecha2.mes)) ||
    ((fecha1.anyo == fecha2.anyo) && (fecha1.mes == fecha2.mes) && (fecha1.dia < fecha2.dia)))
    printf("%d %d %d\n", fecha1.dia, fecha1.mes, fecha1.anyo);
else
    printf("%d %d %d\n", fecha2.dia, fecha2.mes, fecha2.anyo);
```



**SOLUCIÓN:**

```
if (fecha1 < fecha2)
    printf("%d %d %d\n", fecha1);
else
    printf("%d %d %d\n", fecha2);
```

NO se pueden comparar dos estructuras directamente.

```
if ((fecha1.anyo < fecha2.anyo) &&
    (fecha1.mes < fecha2.mes) &&
    (fecha1.dia < fecha2.anyo))
    printf("%d %d %d\n", fecha1.dia, fecha1.mes, fecha1.anyo);
else
    printf("%d %d %d\n", fecha2.dia, fecha2.mes, fecha2.anyo);
```

Compara `fecha1.dia` con `fecha2.anyo` lo cual no tiene sentido.

La respuesta correcta es:

```
if ((fecha1.anyo < fecha2.anyo) ||
    ((fecha1.anyo == fecha2.anyo) && (fecha1.mes < fecha2.mes)) ||
    ((fecha1.anyo == fecha2.anyo) && (fecha1.mes == fecha2.mes) && (fecha1.dia <
    fecha2.dia)))
    printf("%d %d %d\n", fecha1.dia, fecha1.mes, fecha1.anyo);
else
    printf("%d %d %d\n", fecha2.dia, fecha2.mes, fecha2.anyo);
```

Pregunta 3

Correcta

Puntúa 1,00
sobre 1,00

¿Qué muestra por pantalla el siguiente programa?

```
#include <stdio.h>
void funcion1 (int x)
{
    x--;
}
void funcion2 (int * p)
{
    (*p)++;
    funcion1 (*p);
}
void funcion3 (int * p)
{
    (*p)++;
    p++;
    funcion2 (p);
}
int main ()
{
    int x = 20, y = 22;
    funcion3 (&x);
    printf ("%d", x);
    return 0;
}
```

Seleccione una:

- ☒ 21 ✓
- ☐ 23
- ☐ 22

EXPLICACIÓN:

funcion3 aumenta el valor de la variable x a 21. Luego modifica el puntero, por lo que p ya no apunta a la variable original, de modo que las siguientes modificaciones no se realizan sobre x.

La respuesta correcta es: 21

Pregunta 4

Correcta

Puntúa 1,00
sobre 1,00

Dado el siguiente tipo abstracto de dato Usuario

```
typedef struct {  
    char nombre [20];  
    char contra [20];  
    Fecha fecha;  
} Usuario;
```

Se desea declarar una variable de tipo "Usuario". Cuál de las siguientes declaraciones es correcta:

Seleccione una:

- ☒ Usuario users; ✓
- ☐ usuario usuario;
- ☐ Usuario nombre user;

SOLUCIÓN:

Las respuestas erróneas utilizan usuario en minúscula para el tipo de dato y dan dos nombres (nombre y user) a dos variables que no separan por comas.

La respuesta correcta es: Usuario users;

Pregunta 5

Incorrecta

Puntúa -0,50
sobre 1,00

Mi proyecto está escrito íntegramente en el fichero main.c. La siguiente instrucción

```
gcc main.c
```

genera ...

Seleccione una:

- ☐ un fichero ejecutable de nombre a.out
- ☐ un fichero objeto de nombre main.o
- ☒ un fichero ejecutable de nombre main ✗

EXPLICACIÓN:

La instrucción gcc sin modificadores genera un ejecutable de nombre a.out.

Para cambiar el nombre del ejecutable hemos de utilizar el modificador -o. Por ejemplo, con la siguiente instrucción, el ejecutable recibe el nombre "resultado":

```
gcc -o resultado main.c
```

La respuesta correcta es: un fichero ejecutable de nombre a.out

Pregunta 6

Correcta

Puntúa 1,00
sobre 1,00

¿Qué escribe el siguiente programa?

```
#include <stdio.h>
int main ()
{
    int i;

    for (i=1; i<4; i++) {
        if (i%2)
            printf ("%d", i);
    }
    return 0;
}
```

Seleccione una:

- ☐ 24
- ☒ 13 ✓
- ☐ 1

SOLUCIÓN:

Recuerda que `if (i%2)` es equivalente a `if (i%2 != 0)`.

- La variable `i` se inicializa a 1.
- Como `1<4` es cierto, se ejecuta el cuerpo del bucle:
 - Como `1%2` es 1, es decir, cierto, se escribe `i`, es decir, 1.
 - Se incrementa `i`, que pasa a valer 2.
- Como `2<4` es cierto, se ejecuta el cuerpo del bucle:
 - Como `2%2` es 0, es decir, falso, no se escribe el valor de `i`.
 - Se incrementa `i`, que pasa a valer 3.
- Como `3<4` es cierto, se ejecuta el cuerpo del bucle:
 - Como `3%2` es 1, es decir, cierto, se escribe `i`, es decir 3.
 - Se incrementa `i`, que pasa a valer 4.
- Como `4<4` es falso, se termina el bucle.

Por lo tanto, el programa escribe 1 3.

La respuesta correcta es: 13

Pregunta 7

Correcta

Puntúa 1,00
sobre 1,00

Dado el siguiente programa, ¿cuál sería el prototipo de la función que calcula la nota media y cómo se llamaría a este método ?

```
#include <stdio.h>
// Prototipo de la funcion
{
    double m = 0;
    int i;
    for (i = 0; i < numNotas; i++)
        m += notas[i];
    m = m / (numNotas * 1.0);
    return m;
}
int main() {
    double notas[] = {9.8, 6.5, 7, 8.3, 5};
    int numNotas = 5;
    double resultado;
    //Llamada a la función para calcular la nota media
    printf("Nota Media = %.2lf\n", resultado);
    return 0;
}
```

Seleccione una:



Prototipo:

double media(double notas)

Llamada a la función:

resultado = media(notas);



Prototipo:

void media(double *notas, int numNotas)

Llamada a la función:

resultado = media(notas);



Prototipo:

double media(double *notas, int numNotas)

Llamada a la función:

resultado = media(notas, numNotas);



SOLUCIÓN:

Prototipo: `double media(double notas)` y Llamada a la función: `resultado = media(notas);` no es correcta pues el prototipo de la función tiene un solo argumento que además es una variable, no una tabla.

Prototipo: `void media(double *notas, int numNotas)` y Llamada a la función: `resultado = media(notas);` no es correcta pues la llamada a la función se hace con sólo un argumento.

La respuesta correcta es:

Prototipo:

```
double media(double *notas, int numNotas)
```

Llamada a la función:

```
resultado = media(notas, numNotas);
```

Pregunta 8

Correcta

Puntúa 1,00
sobre 1,00

Queremos mostrar por pantalla 3 números, uno en cada línea, alineados a la derecha de forma que el número más grande que se pueda mostrar (número máximo de cifras = 7) quede a 2 espacios del borde izquierdo. ¿Cuál sería la sentencia correcta para mostrar lo deseado?

Seleccione una:



```
printf("%9d\n%9d\n%9d\n", num1, num2, num3);
```



```
printf("%7d\n%7d\n%7d\n", num1, num2, num3);
```



```
printf("%7d\n7%d\n7%d\n", num1, num2, num3);
```

Respuesta correcta

SOLUCIÓN:

El número máximo de cifras de un número es 7, si queremos que un número con esas cifras esté separado 2 espacios del borde izquierdo (y alineado a la derecha) lo que realmente queremos es alinear los números con una anchura de 9 espacios.

La respuesta correcta es:

```
printf("%9d\n%9d\n%9d\n", num1, num2, num3);
```



Pregunta 9

Correcta


Puntúa 1,00
sobre 1,00

¿Qué imprimiría el siguiente código por pantalla?

```
#include<stdio.h>

int main(){
printf("%%%\n%%");
return 0;
}
```

Seleccione una:

- ☐ Dos líneas, cada una con cuatro símbolos %
- ☐ Un línea con cuatro símbolos %
- ☒ Dos líneas, cada una con dos símbolos % 

SOLUCIÓN:

Por cada dos % en un printf, se imprime sólo uno por pantalla

La respuesta correcta es: Dos líneas, cada una con dos símbolos %

Pregunta 10

Correcta

Puntúa 1,00
sobre 1,00

La siguiente función debe reservar una cadena de caracteres de dimensión igual al valor entero pasado como argumento. La función debe devolver la dirección de la tabla o NULL si no se pudo llevar a cabo la reserva. ¿Cuál de las siguientes versiones es correcta?

Versión 1:

```
char * func (int n){
    char * cad;

    cad = (char *) malloc (n * sizeof (char));

    if (!cad)
        return cad;
    else
        return NULL;
}
```

Versión 2:

```
char * func (int n){
    char * cad;

    cad = (char *) malloc (n * sizeof (char));

    if (cad)
        return cad;
    else
        return NULL;
}
```

Versión 3:

```
char * func (int n){
    char * cad;

    cad = (char *) malloc (n * sizeof (char));

    if (cad == NULL)
        return cad;
    else
        return NULL;
}
```

Seleccione una:

- ☐ Versión 1
- ☐ Versión 3
- ☒ Versión 2 ✓

EXPLICACIÓN:

Las versiones 1 y 3 coinciden porque

```
if (!cad)
    return cad;
else
    return NULL;
```

es equivalente a

```
if (cad == NULL)
    return cad;
else
    return NULL;
```

Ambas devuelven NULL tanto si la reserva tuvo éxito como si no, es decir, no cumplen con los requerimientos.

La respuesta correcta es: Versión 2

Pregunta 11

Correcta

Puntúa 1,00
sobre 1,00

¿Cuántas líneas tiene el fichero "prueba.txt" después de ejecutar este programa?

```
int main()
{
    FILE *f;
    int i;

    f=fopen("prueba.txt","w");

    for (i=1; i<100; i=i*2)
    {
        fprintf(f,"%d\n",i);
    }

    fclose(f);
    return 0;
}
```

Seleccione una:

- ☒ 8 ✓
- ☐ Ninguna
- ☐ 100

Cada fprintf imprime una línea, más la línea final.

La respuesta correcta es: 8

Pregunta 12

Correcta

Puntúa 1,00
sobre 1,00

Estudia las siguientes declaraciones:

```
typedef struct {  
    int c;  
} H;
```

```
typedef struct {  
    int a;  
} N;
```

```
typedef struct {  
    int y;  
    N b;  
    H s;  
} V;
```

```
typedef struct {  
    int z;  
    V c;  
} M;
```

```
typedef struct {  
    V x;  
    M a;  
    V r;  
} U;
```

U mi;

¿Cuál de los siguientes accesos es correcto?

Seleccione una:

- ☐ mi.a.c.x
- ☒ mi.x.b.a ✓
- ☐ mi.r.y.c

mi es de tipo U

En mi, dentro de la estructura U, se puede referenciar al miembro x, que es de tipo V

En x, dentro de la estructura V, se puede referenciar al miembro b, que es de tipo N

En b, dentro de la estructura N, se puede referenciar al miembro a.

La respuesta correcta es: mi.x.b.a

Pregunta 13

Correcta

Puntúa 1,00
sobre 1,00

¿Qué escribe el siguiente programa?

```
#include <stdio.h>

int f (int a) {
    if (a%2)
        return 0;
    else if (a%3)
        return 1;
    return 2;
}

int main () {
    int m;
    m = f(6);
    printf ("%d", m);
```

```
    return 0;
}
```

Seleccione una:

- ☒ 2 ✓
- ☐ 1
- ☐ 0

EXPLICACIÓN:

La instrucción

```
if (a%2) ...
```

es equivalente a

```
if (a%2 != 0) ...
```

Estudiemos la llamada f(6). Para a=6, tanto a%2 como a%3 vale 0. Por lo tanto, se ejecuta la instrucción return 2;. Es decir, f devuelve 2, con lo que m recibe el valor 2.

La respuesta correcta es: 2

Pregunta 14

Correcta

Puntúa 1,00
sobre 1,00

¿Qué se imprime por pantalla al ejecutar el siguiente código?

```
char func1(char c) {
```

```
    return 'c';
```


```
}
```

```
int main() {
```

```
    printf("%c", func1('f'));  
    return 0;
```

```
}
```

Seleccione una:

- ☐ f
- ☐ cf
- ☒ c 

EXPLICACIÓN:

Se imprime lo que devuelve la función func1, esto es, el carácter c.

La respuesta correcta es: c

Pregunta 15

Correcta

Puntúa 1,00
sobre 1,00

¿Qué muestra por pantalla el siguiente programa?

```
#include <stdio.h>
void f2 (int a, int *b)
{
    a++;
    *b=*b+1;
    a=a+a;
    *b=*b+*b;
    printf(" %d %d ",a,*b);
}
int main()
{
    int x=0,y=0;
    f2(x,&y);
    printf(" %d %d\n",x,y);
    return 0;
}
```

Seleccione una:

- ☒ 2 2 0 2 ✓
- ☐ 2 1 0 1
- ☐ 2 2 0 0

EXPLICACIÓN:

Dentro de la función a pasa tomar el valor de x y b el valor de *y.
Ambas se incrementan dos veces por lo que dentro de la función se imprime por pantalla 2 2.
Al salir de la función sólo se tiene en cuenta la modificación de la variable y (que es la única que se ha pasado por referencia) por lo que se muestra 0 2 por pantalla.

La respuesta correcta es: 2 2 0 2

Pregunta 16

Correcta

Puntúa 1,00
sobre 1,00

¿Qué mostrará por pantalla el siguiente programa?

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

```
int main() {  
    int i;
```

```
    for (i = 70; i; i -= 2) {  
        printf("%d\n", i);  
        i -= 2;  
    }
```

```
    return 0;  
}
```

Seleccione una:

- ☐ Se imprimen todos los números del 70 al 2 de dos en dos.
- ☐ Se imprimen todos los números del 70 al 2 de cuatro en cuatro.
- ☒ Se entra en un bucle infinito. ✓

Respuesta correcta

SOLUCIÓN:

tras cada iteración del bucle for i decrementa su valor 4 unidades:

iteración 1: i = 66

iteración 2: i = 62

...

iteración 17: 2

como 70 no es divisible por 4 i nunca llegará a valer 0 exactamente, por lo que la condición del bucle no se cumplirá y se creará un bucle infinito.

La respuesta correcta es: Se entra en un bucle infinito.

Pregunta 17

Correcta

Puntúa 1,00
sobre 1,00

¿Cuántas veces se mostrará el menú de inicio antes de terminar el programa si el usuario teclea como opción 3?

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

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

```
int main() {
```

```
int opcion;
```

```
do  
{
```

```
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");
```

```
}
```

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

```
return 0;
```

```
}
```

Seleccione una:

- ☒ Una. ✓
- ☐ Ninguna.
- ☐ Hasta que el usuario vuelva a introducir '3'.

SOLUCIÓN:

Al usuario se le muestra una sólo vez el menú, dado que la opción 3 se corresponde con `salir` (fíjate en la posición que tiene en el enumerado y el valor al que se inicializa `altaUsuario`). La condición de mantenimiento del bucle es que `opcion != salir`, así al tomar el valor `salir` termina.

La respuesta correcta es: Una.

Pregunta 18

Correcta

Puntúa 1,00
sobre 1,00

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

```
#define TAM 100
```

```
typedef struct{
```

```
    int sumandos[TAM];
    int numSumandos;
    int suma;
```

```
}Suma;
```

```
//Prototipo de la función
```

```
{
```

```
    int i;
```

```
    s->suma=0;
```

```
    for (i=0;i<s->numSumandos;i++)
        s->suma+=s->sumandos[i];
```

```
}
```

```
int main() {
```



```
Suma suma;
```

```
suma.sumandos[0]=1;  
suma.sumandos[1]=2;  
suma.sumandos[2]=3;  
suma.sumandos[3]=4;  
suma.sumandos[4]=5;  
suma.numSumandos=5;
```

```
//Llamada a la función
```

```
printf("%d", suma.suma);
```

```
return 0;
```

```
}
```

Elige el prototipo y la llamada de la función acordes con el código anterior.

Seleccione una:

- ☐ Prototipo: void sumar(Suma s); Llamada: sumar(suma);
- ☒ Prototipo: void sumar(Suma *s); Llamada: sumar(&suma); ✓
- ☐ Prototipo: void sumar(Suma *s); Llamada: sumar(*suma);

EXPLICACIÓN:

La función debe recibir un puntero a la estructura para que se modifique el valor de sus campos. Una pista extra de que debe funcionar así es que dentro de la función accedemos a los campos con `->` en lugar de `.`, por lo que debemos estar trabajando con un puntero.

En la llamada a la función, puesto que tenemos declarada una variable de tipo Suma, deberemos pasar su dirección de memoria usando **&**.

La respuesta correcta es: Prototipo: void sumar(Suma *s); Llamada: sumar(&suma);

Pregunta 19

Correcta

Puntúa 1,00
sobre 1,00

Dada la siguiente definición de una cadena de caracteres

`char fecha[DIM];`

¿Cuál sería el tamaño mínimo del defide DIM para almacenar una fecha en formato `yyyymmdd`, como por ejemplo 20140912?

Seleccione una:

☐ 8

☒ 9 ✓

☐ 10

Respuesta correcta

El tamaño de una cadena debe de incluir el carácter fin de cadena 0 por tanto es uno más del tamaño necesario

La respuesta correcta es: 9

Pregunta 20

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:

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

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

Volver a: General ➡