

PROGRAMACIÓN I

Comenzado el sábado, 15 de diciembre de 2018, 13:09

Estado Finalizado

Finalizado en sábado, 15 de diciembre de 2018, 13:27

Tiempo empleado 18 minutos 5 segundos

Puntos 14,00/20,00

Calificación 7,00 de 10,00 (70%)

Pregunta 1

Correcta

Puntúa 1,00
sobre 1,00

Dada la siguiente declaración:

```
const int meses[12] = {31,28,31,30,31,30,31,31,30,31,30,31};
```

¿Cuál de las siguientes sentencias produce un error?

Seleccione una:

- ☒ `scanf("%d", &meses[3])` ✓
- ☐ `printf("%d", meses[3] + meses[4]);`
- ☐ `printf("%d", meses[3]);`

SOLUCIÓN:

meses se ha declarado como una contante (mediante el modificador const) por lo que su valor no se puede cambiar ni mediante una asignación ni mediante un scanf, como en este caso.

La respuesta correcta es: `scanf("%d", &meses[3])`

Pregunta 2

Correcta

Puntúa 1,00
sobre 1,00

El siguiente programa provoca un error durante su compilación. Para resolverlo hemos de ...

```
#include <stdio.h>

int main (){

    printf ("%lf",dividir_por_dos(5));

    return 0;

}

double dividir_por_dos (int x) {

    return x*0.5;

}
```

Seleccione una:

- ☒ declarar la función dividir_por_dos antes de main ✓
- ☐ cambiar la definición de dividir_por_dos para que devuelva un int
- ☐ ninguna de las otras

EXPLICACIÓN:

Este programa genera avisos y errores de compilación porque en la llamada a dividir_por_dos, el compilador supone que la función devuelve un int. Esto entra en conflicto con el formato de la instrucción printf. Además, posteriormente, en la definición de dividir_por_dos, el compilador descubre que el tipo devuelto es double, lo cual entra en conflicto con la suposición que hace anteriormente.

Si cambiamos el tipo de dato devuelto por la función, también tendremos que cambiar el cuerpo de la función o de lo contrario se generarán nuevos errores.

La solución consiste en colocar la declaración de la función antes de la llamada como se ve a continuación:

```
#include <stdio.h>
double dividir_por_dos (int x);
int main (){
    printf ("%lf",dividir_por_dos(5));
    return 1;
}
double dividir_por_dos (int x) {
    return x*0.5;
}
```

Ahora, el compilador ya sabe qué devuelve la función y qué argumentos se le han de pasar, antes de llamarla.

La respuesta correcta es: declarar la función dividir_por_dos antes de main

Pregunta 3

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 terminar la ejecución del programa
- ☐ para saltar las instrucciones que aparezcan hasta la siguiente llave de cierre }
- ☒ 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 4

Incorrecta

Puntúa -0,50
sobre 1,00

¿Cuál de las instrucciones marcadas 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:

- ☐ 1
- ☐ 3
- ☒ 2 

EXPLICACIÓN:

- La instrucción 2 no es correcta porque `*q->codigo` debería ser `(*q)->codigo` o `q->codigo`.
- La instrucción 3 no es correcta porque `&prov1` es la dirección de una `Provincia` y la función espera una variable de tipo `Provincia`.

La respuesta correcta es: 1

Pregunta 5

Correcta

Puntúa 1,00
sobre 1,00

La función duplicar debe crear una copia de la cadena pasada como argumento y devolver su dirección. ¿Cuál de las siguientes versiones es correcta?

Versión 1:

```
char * duplicar (char * s){
    char * p;
    char * q=s;
    int n;

    for (n=0; *s; n++, s++);

    p = (char *) malloc (n * sizeof (char));
    strcpy(p,q);
    return p;
}
```

Versión 2:

```
char * duplicar (char * s){
    char * p;
    char * q=s;
    int n;

    for (n=0; *s; n++, s++);

    p = (char *) malloc ((n+1) * sizeof (char));
    strcpy(q,p);
    return p;
}
```

Versión 3:

```
char * duplicar (char * s){
    char * p;
    char * q=s;
    int n;

    for (n=0; *s; n++, s++);

    p = (char *) malloc ((n+1) * sizeof (char));
    strcpy(p,q);
    return p;
}
```

Seleccione una:

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

^

EXPLICACIÓN:

- La versión 1 no es correcta porque reserva n caracteres y debería reservar n+1. El carácter extra sirve para alojar el final de cadena.

- La versión 2 no es correcta porque copia la cadena p en la q, cuando debería copiar la q en la p. Fíjate en que q apunta al principio de la cadena pasada como argumento.

La respuesta correcta es: Versión 3

Pregunta 6

Correcta

Puntuía 1,00
sobre 1,00

¿Qué se imprime por pantalla al ejecutar este programa?

```
int main()
{
    FILE *f;
    int i, a, b;

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

    for (i=1; i<10; i++)
    {
        fprintf(f, "%d %d\n", i, i*i);
    }

    fclose(f);

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

    fscanf(f, "%d %d\n", &a, &b);
    fscanf(f, "%d %d\n", &a, &b);
    fscanf(f, "%d %d\n", &a, &b);

    printf("%d %d", a, b);

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

Seleccione una:

- ☒ 3 9 ✓
- ☐ 1 2 3
- ☐ Nada, da un error

El primer bucle imprime nueve lineas, desde 1 1 hasta 9 81, luego se leen las tres primeras, y se imprime la última leída (la tercera), es decir los números 3 9

La respuesta correcta es: 3 9

Pregunta 7

Correcta

Puntúa 1,00
sobre 1,00

Después de ejecutar la siguiente instrucción

```
int datos [5] = {20,39,48,22,1};
```

¿qué valor tendrá la componente datos[3-1] de la tabla?

Seleccione una:

- ☐ 39
- ☐ 22
- ☒ 48 ✓

SOLUCIÓN:

datos[3-1] es igual a datos[2], que corresponde con el valor de la tercera posición de la tabla: 48

La respuesta correcta es: 48

Pregunta 8

Correcta

Puntúa 1,00
sobre 1,00

La función `buscarPorFechaNacimiento` tiene la siguiente cabecera:

```
int buscarPorFechaNacimiento (Sesion sesion, int anyo, int * indices);
```

Dadas las siguientes declaraciones en `main` (`U` es la macro que contiene el número máximo de usuarios de tuenti):

```
Sesion ses;  
int anyo, num;  
int indices[U];
```

¿Cuál de las siguientes llamadas a la función es correcta?

```
num = buscarPorFechaNacimiento (ses, anyo, &indices); // 1  
num = buscarPorFechaNacimiento (&ses, anyo, indices); // 2  
num = buscarPorFechaNacimiento (ses, anyo, indices); // 3
```

Seleccione una:

- ☒ 3 ✓
- ☐ 2
- ☐ 1

EXPLICACIÓN:

- La llamada 1 es incorrecta porque el tercer argumento es la dirección del nombre de la tabla de enteros y la función espera la dirección de un entero, como `indices`.
- La llamada 2 es incorrecta porque el primer argumento es la dirección de `ses` y la función espera una variable de tipo `Sesion`, no su dirección.

La respuesta correcta es: 3



Pregunta 9

Incorrecta

Puntúa -0,50
sobre 1,00

¿Qué muestra el programa (más abajo) si el usuario teclea lo siguiente cuando se le pregunta?

Este es el test de Programacion 1 numero 8

e

Código del programa:

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

```
#define DIM 100
```

```
int contarCaracteres(char *cadena, char caracter){  
    int cuenta;  
    for(cuenta=0;*cadena;cadena++){  
        if(*cadena==caracter)  
            cuenta++;  
    }  
    return cuenta;  
}  
  
int main () {  
  
    char cadena[DIM];  
    char caracter;  
  
    printf("Indroduzca una cadena: ");  
    gets(cadena);  
  
    printf("Introduzca un caracter: ");  
    scanf("%c", &caracter);  
  
    printf("La cadena %s tiene %d caracteres %c",  
        cadena, contarCaracteres(cadena, caracter), caracter);
```

```
    return 0;  
}
```

Seleccione una:



La cadena "Este es el test de Programacion 1 numero 8" tiene 6 caracteres 'e'



La cadena "Este es el test de Programacion 1 numero 8" tiene <valor indeterminado> caracteres 'e'



^

La cadena "Este es el test de Programacion 1 numero 8" tiene 7 caracteres 'e'



Respuesta incorrecta.

SOLUCIÓN:

El programa cuenta el número de veces que aparece el carácter que introduce el usuario en la cadena que se la ha pedido previamente. Para ello, recorre la cadena, mediante el puntero a char `p`, y compara cada carácter con el que ha introducido el usuario ('e' en este caso). Si coinciden, incrementa la cuenta. Esta comparación distingue entre mayúsculas y minúsculas, por lo tanto 'e' y 'E' son caracteres diferentes.

Finalmente, se muestra un mensaje informando del resultado. En este ejemplo se mostraría el siguiente texto:

La cadena "Este es el test de Programacion 1 numero 8" tiene 6 caracteres 'e'

La respuesta correcta es:

La cadena "Este es el test de Programacion 1 numero 8" tiene 6 caracteres 'e'

Pregunta 10

Correcta

Puntúa 1,00
sobre 1,00

¿Cuál será la forma correcta de llamar a la función `buscarPorNombre`, si declaramos la variable `nombre` de la siguiente forma

```
char *nombre;
```

y la función tiene la siguiente cabecera?

```
int buscarPorNombre(char *nombre)
```

Seleccione una:

- ☐ `num=buscarPorNombre(&nombre);`
- ☒ `num=buscarPorNombre(nombre);` ✓
- ☐ `num=buscarPorNombre(*nombre);`

EXPLICACIÓN:

`nombre` es un puntero (equivale a `&nombre[0]`), por lo que es lo que se le ha de pasar a la función

La respuesta correcta es: `num=buscarPorNombre(nombre);`

Pregunta 11

Correcta

Puntúa 1,00
sobre 1,00

Dados los siguientes códigos

1)

```
int *p,x;  
*p=5;
```

2)

```
int *p,x;  
p=&x;  
*p=5;
```

¿Cuál de los 2 es correcto?

Seleccione una:

- ☐ ambas opciones
- ☐ opción 1
- ☒ opcion 2 ✓

EXPLICACIÓN:

El primer código no es correcto pues p es un puntero que no apunta a ninguna zona de memoria, por lo que no se le puede asignar ningún valor a su contenido.

La respuesta correcta es: opcion 2

Pregunta 12

Correcta

Puntúa 1,00
sobre 1,00

En un programa en C se utiliza el siguiente tipo para almacenar fechas:

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

A lo largo de un programa que utiliza dos variables de tipo *Fecha*, *fecha1* y *fecha2*, aparece la siguiente sentencia *if* seguida de una llamada a la función *printf()*:

```
if ((fecha1.anno >= 0) && (fecha2.anno >= 0) && !(fecha1.anno - fecha2.anno))  
    printf("%s\n", mensaje);
```

¿Qué debería contener el string *mensaje* para describir las fechas comparadas?

Seleccione una:

- ☒ "Los años son iguales" ✓
- ☐ "Los años no son correctos"
- ☐ "Los años son distintos"

Respuesta correcta

El operador lógico ! devuelve TRUE si la expresión aritmética a la que se aplica vale 0 y devuelve FALSE en otro caso.

La respuesta correcta es: "Los años son iguales"

Pregunta 13

Correcta

Puntúa 1,00
sobre 1,00

¿Cuál es la biblioteca básica a importar, para cualquier programa escrito en C?

Seleccione una:

- ☒ `stdio.h` ✓
- ☐ `studio.h`
- ☐ `conio.h`

SOLUCIÓN:

En la biblioteca `stdio.h` incluye, entre otras, la definición de las funciones `scanf` y `printf`.

La respuesta correcta es: `stdio.h`

Pregunta 14

Correcta

Puntúa 1,00
sobre 1,00

¿Cuál de los siguientes códigos es correcto?

Seleccione una:



```
const int diasDelMes[12] = {31,28,31,30,31,30,31,31,30,31,30,31};
int mes;
printf("Mes del anyo?");
scanf("%d", &mes);
printf("El mes %d tiene %d días", mes, diasDelMes[mes-1]);
```



```
const int diasDelMes[12] = {31,28,31,30,31,30,31,31,30,31,30,31};
int mes;
printf("Mes del anyo?");
scanf("%d", &mes);
printf("El mes %d tiene %d días", mes, diasDelMes[mes+1]);
```



```
const int diasDelMes[12] = {31,28,31,30,31,30,31,31,30,31,30,31};
int mes;
printf("Mes del anyo?");
scanf("%d", &mes);
printf("El mes %d tiene %d días", mes, diasDelMes[mes]);
```

SOLUCIÓN:

La respuesta correcta corresponde con la solución del ejercicio 2 de la práctica 2.

La respuesta correcta es:

```
const int diasDelMes[12] = {31,28,31,30,31,30,31,31,30,31,30,31};
int mes;
printf("Mes del anyo?");
scanf("%d", &mes);
printf("El mes %d tiene %d días", mes, diasDelMes[mes-1]);
```

Pregunta 15

Correcta

Puntúa 1,00
sobre 1,00


¿Qué escribe el siguiente programa?

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

    for (i=2; i>0; i--) {
        printf ("%d", i);
    }
```

```
    return 0;
}
```

Seleccione una:

- ☐ 210
- ☐ 10
- ☒ 21 

SOLUCIÓN:

- La variable i se inicializa a 2.
- Como $2 > 0$ es cierto, se ejecuta el cuerpo del bucle:
 - Se escribe i, es decir, 2.
 - Se decrementa i, que pasa a valer 1.
- Como $1 > 0$ es cierto, se ejecuta el cuerpo del bucle otra vez:
 - Se escribe el valor de i, es decir, 1.
 - Se decrementa i, que pasa a valer 0.
- Como $0 > 0$ es falso, se termina el bucle.

Por lo tanto, el programa escribe 21.

La respuesta correcta es: 21

Pregunta 16

Correcta

Puntúa 1,00
sobre 1,00

Si definimos la función:

```
double areaTriangulo(int base, int altura) {  
    double area;  
    area = base * altura / 2.0;  
    return area;  
}
```

¿Cómo debemos llamar a esta función dentro del siguiente programa para que pinte el área por pantalla?

```
int main() {  
    int ba, al;  
    double ar;  
    printf("Introduce la base del triángulo: ");  
    scanf("%d", &ba);  
    printf("Introduce la altura del triángulo: ");  
    scanf("%d", &al);  
    //Llamada a la función  
    printf("El área del triángulo de base %d y altura %d es %.2lf", ba, al, a  
r);  
    return 0;  
}
```

Seleccione una:

- ☐ areaTriangulo();
- ☒ ar=areaTriangulo(ba, al); ✓
- ☐ ar=areaTriangulo();

EXPLICACIÓN:

La función necesita dos argumentos de tipo int para ejecutarse y devuelve un valor de tipo double. Por lo tanto, la llamada correcta es:

```
ar = areaTriangulo (ba, al);
```

La respuesta correcta es: ar=areaTriangulo(ba, al);

Pregunta 17

Incorrecta


Puntúa -0,50
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 
- ☐ 1
- ☐ 11

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 18

Correcta

Puntúa 1,00
sobre 1,00

¿Cuál es la diferencia entre la sentencia while() y la sentencia do while()?

Seleccione una:

- ☐ La sentencia do while() es más segura, el programa es más difícil que se quede en un bucle infinito.
- ☐ La sentencia do while () es mas legible, los programas quedan mas sencillos de entender.
- ☒ En la sentencia do while() siempre se ejecuta al menos una vez el cuerpo del bucle. 

La respuesta correcta es: En la sentencia do while() siempre se ejecuta al menos una vez el cuerpo del bucle.

Pregunta 19

Incorrecta

Puntuación -0,50
sobre 1,00

El siguiente programa escrito en C no funciona bien siempre que se ejecuta:

```
#define DIM 64
```

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

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

```
Contacto* iniContacto(Contacto* cnt, char* nmb, char* apII, int dn,  
int mn, int an){  
    Fecha * fn= NULL;  
  
    fn = (Fecha*)malloc(sizeof(Fecha));  
    if (!fn) return NULL;
```

```
    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;  
  
    free(fn);
```

```
    return cnt;  
}
```

```
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* amigo= NULL;
```

```
amigo= (Contacto*) malloc(sizeof(Contacto));  
if (!iniContacto(amigo, "Manuel", "Molina", 3, 3, 1999)) {  
    printf("Error al reservar memoria para amigo.\n");  
    return 1;  
}
```

```
impContacto(amigo);
```

```
free(amigo);
```

```
return 0;
```

```
}
```

¿Cuál de las siguientes afirmaciones es correcta?

Seleccione una:

- ☐ La variable *fn* es local a la función *iniContacto()*, por lo que no es adecuado utilizarla para inicializar el campo *fecha_nacimiento* de *amigo*.
- ☐ La memoria reservada para *amigo->fecha_nacimiento* no hay que liberarla dentro de la función *iniContacto()*, sino justo antes de liberar la memoria apuntada por *amigo* en *main()*.
- ☒ Eliminando la línea "*free(fn);*" del código de la función *iniContacto()* ya no hay ningún problema. ❌

Respuesta incorrecta.

El problema es que se libera la memoria reservada para el campo *fecha_nacimiento* dentro de la función *iniContacto()*, por lo que el espacio de memoria al que apunta *amigo->fecha_nacimiento* en *main()* puede reutilizarse para otra cosa.

La respuesta correcta es: La memoria reservada para *amigo->fecha_nacimiento* no hay que liberarla dentro de la función *iniContacto()*, sino justo antes de liberar la memoria apuntada por *amigo* en *main()*.

Pregunta 20

Correcta

Puntúa 1,00
sobre 1,00

¿Qué comandos habría que añadir para que el siguiente programa calculase si un número es primo o no y lo mostrase por pantalla?

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int num;
    int divisible = 0;
    int i;
    printf("Introduzca un numero: ");
    scanf("%d", &num);

    // Comandos para calcular si es primo
```

```
    if (divisible)
        printf("No es primo\n");
    else
        printf("Es primo\n");

    return 0;
}
```

Selecione una:

☐

```
    for (i = 2; i < num; i++) {
        if (num % i==0) {
            divisible = 0;
            break;
        }else {
            divisible = 1;
            break;
        }
    }
}
```

☐

```
    for (i = 0; i < num; i++) {
        if (num % i==0) {
            divisible = 1;
        }
    }
}
```

☐

```
for (i = 2; i < num; i++) {  
    if (num % i==0) {  
        divisible = 1;  
        break;  
    }  
}
```



SOLUCIÓN:

Para que un número sea primo, sólo debe ser divisible por 1 y por sí mismo. Por lo tanto, si un número es divisible por otro (diferente a él mismo y 1), entonces, no será primo. De esta forma, el código para saber si un número es primo o no debe recorrer todos los números menores a él, comenzando desde 2 (se da por hecho que es divisible por 1). Si en algún momento del bucle el número que saber si es primo es divisible por el número que se está probando, podemos terminar el bucle y afirmar que el número introducido no es primo.

La respuesta correcta es:

```
for (i = 2; i < num; i++) {  
    if (num % i==0) {  
        divisible = 1;  
        break;  
    }  
}
```

[Volver a: General ➡](#)

