Programación

Test Autoevaluación
Tema 5

Autores:

M. Paz Sesmero Lorente
Paula de Toledo Heras
Fco. Javier Ordoñez Morales
Juan Gómez Romero
Jose A. Iglesias Martínez
Jose Luis Mira

Universidad Carlos III de Madrid
www.uc3m.es

SOLUCIONES

1. **¿Cuál es el resultado del siguiente programa?**

```c
#include <stdio.h>

void imprimeDatos(int b, int a);

int main(void){
  int a, b;
  a= 5;
  b = 10;
  imprimeDatos(5,10);
  return 0;
}
void imprimeDatos(int b, int a){
    printf("VALORES: %d y %d", a, b);
    return;
}
```

Solución: La correspondencia entre los parámetros reales (llamada) y los parámetros formales (definición) de la función establece que al inicio de la función b=5 y a=10. Por tanto la sentencia `printf("VALORES: %d y %d", a, b);` mostrará por pantalla VALORES 10 y 5.

   a. **VALORES: 10 y 5. @**
   b. **VALORES: 5 y 10.**
   c. **El programa no compila porque utiliza variables globales.**
   d. **El programa no compila porque la llamada a la función imprimeDatos es incorrecta.**

2. **Señale la forma correcta de terminar la frase: "En el programa principal, el valor de una variable que se pasa como parámetro por valor a una función**
   a. **no puede ser modificado.** Falso. Los parámetros de una función son variables declaradas en otra función (que puede ser la función main). Por tanto, y al igual que cualquier otra variable, su valor se puede modificar.
   b. **cambia cuando dentro de la función se modifica el valor de dicho parámetro.** Falso. Cuando una variable se pasa por valor las modificaciones realizadas dentro de la función no tienen efecto fuera de ella.
   c. **no cambia aunque dentro de la función se modifique el valor de dicho parámetro.** Verdadero.
   d. **no puede conocerse después de la llamada, puesto que la variable se destruye**. Falso. Son las variables locales de la función las que se destruyen tras su ejecución.

3. **Dado el siguiente programa: ¿cuál de las siguientes líneas de código podría escribirse en la línea de guiones (-----) para que el programa compilase sin errores?**

```
#include <stdio.h>
float mi_funcion(float a, int b, char c);

int main()
{
    float num_a=5.5;
    int num_b=4.0;
    char letra;
     -------------------
    return(0);
}

float mi_funcion(float a, int b, char c)
{
    c='k';
    return(a+b);
}
```

   a. **printf ("%f", mi_funcion(num_a,num_b,letra));** Correcto. `mi_funcion` debe recibir tres variables del tipo **`float`**, **`int`**, **`char`** en este orden y devuelve un valor de tipo float. Observando la declaración de `num_a, num_b y letra`, se puede comprobar que estas variables son del tipo indicado
   b. **printf ("%f", mi_funcion(num_b,num_a,letra));** Correcto. En este caso y aunque `num_b, num_a y letra` son, respectivamente, variables de tipo **`int`**, **`float`**, **`char`** al establecer la correspondencia entre parámetros reales (llamada) y parámetros formales se realiza una conversión implícita de tipos (ver tema 3). En este caso: `a=(float)num_b=4.0` (se añade parte decimal nula) y `b=(int)num_a=5` (se trunca la parte decimal) por lo que el valor de retorno `(a+b);` será `9.0`.
   c. **El programa no compila porque la sentencia de asignación: `int num_b=4.0;` es incorrecta.** Falso. En esta sentencia hay una conversión implícita de tipos por lo que es equivalente a **`int num_b=4`**
   d. **Cualquiera de las dos primeras** Correcto

4. **Dado el siguiente programa, señale la afirmación correcta**.

```
#include <stdio.h>
int d (int e,int f);

int main(void)
{     int a,b,c;
      c=d(a,b);
      printf ("%i" , c);
      return 0;
}

int d (int e,int f)
{
      int g,h;
      return (e+g+f+h);
}
```

a. **a, b, c son variables globales.** Falso. Las variables globales son las que se declaran fuera de las funciones (incluida main).
b. **a, b, c son variables locales.**. Correcto
c. **e, f son parámetros reales.** Falso. Los parámetros reales son los que aparecen en la llamada a una función. Los que aparecen en la definición se denominan parámetros formales
d. **e y f son variables globales.** Falso por las razones indicadas en el punto a.

5. **¿Cuál de las siguientes afirmaciones es la más apropiada para continuar la siguiente frase: "Las funciones ….**
   a. **…no son necesarias para programar y casi ningún programador las usa".** Falso. La adecuada división de un programa en subprogramas (funciones) constituye un aspecto fundamental en el desarrollo de cualquier programa.
   b. **...facilitan la programación, evitando tener que repetir código que se utiliza varias veces en el mismo programa".** Cierto. Las funciones pueden ser utilizadas varias veces en el mismo programa o incluso en distintos programas.
   c. **…siempre deben tener al menos un parámetro de entrada".** Falso. Puede haber funciones que no acepten ningún parámetro.
   d. **... facilitan la programación, al permitir la transformación de los datos".** Falso.

6. **¿Cuál es el resultado que se muestra por pantalla tras ejecutar el siguiente programa?**

```
#include <stdio.h>
int producto (int a,int b);

int main(void)
{     int n1=5, n2=7;
      n2 = producto(n1,n2);
      printf ("%i, %i", n2, n1);
      return 0;
}

int producto (int a,int b)
{
      a= 20;
      return (a * b);
}
```

Solución: Al inicio de la función producto, a=5, b=7. Sin embargo, la sentencia `a= 20` hace que el valor devuelto por la función sea (20*7)=140. Por tanto, tras ejecutar `n2 = producto(n1,n2);` n2 toma el valor 140 mientras que n1 sigue valiendo 5 (la modificación realizada dentro de la función no se ve reflejada en el programa principal). Por tanto, los valores mostrados por pantalla serán 140 (b) y 5 (a).

a. **140, 5@**
b. **140, 20**
c. **35, 5**
d. **35, 20**

7. **Señale la respuesta falsa respecto a la técnica de programación modular:**
   a. **Se basa en la descomposición del programa principal en otros más simples.** Cierto.
   b. **Establece que el programa principal solamente puede contener las operaciones fundamentales.** Falso. Cualquier operación fundamental puede implementarse como una función.
   c. **Se basa en el diseño de subprogramas, que pueden reutilizarse en otros programas.** Cierto.
   d. **Permite crear un programa principal más corto y legible.** Cierto.

8. **Señale la afirmación correcta respecto al siguiente programa**

```c
#include <stdio.h>
int Calcular (int a,int b,int s,int r);

int main(void)
{
    int n1=8,n2=2, suma, resta, producto,code;
    code = Calcular (n1,n2,suma,resta);
    printf("%i, %i, %i", suma,resta,producto);
    return 0;
}

int Calcular (int a,int b,int s,int r)
{
    int p;
    s =a+b;
    r=a-b;
    p=a*b;
    return(0);
}
```

   Solución: Para que las operaciones realizadas en la función se vean reflejadas en la función `main` es necesario pasar los parámetros *por referencia*. Puesto que los parámetros usados en la función `Calcular` se pasan *por valor* su valor antes y después de la llamada a la función permanece invariable. Así pues las variables `suma,resta,producto` tendrán un valor indeterminado. Esto indica que la respuesta correcta es la opción b.
   a. **Muestra los resultados correctos para la suma y resta de n1 y n2, pero no el resultado correcto para el producto.**
   b. **No muestra los resultados correctos ni de la suma, ni de la resta ni del producto de n1 y n2@**
   c. **No compila, puesto que los parámetros formales y los reales no tienen el mismo nombre en la llamada y en el prototipo.**
   d. **Utiliza variables globales en la función Calcular**

9. ¿Por qué no deben usarse variables globales dentro de una función?

a. **Porque sus valores solo son accesibles desde el programa main.** Falso. Las variables globales son accesibles desde cualquier función.
b. **Porque dificulta la depuración y mantenimiento del código.** Cierto
c. **Sí se deben usar, siempre que sea necesario acceder a sus valores.** Falso. Siempre hay que evitar el uso de variables globales. Cuando se tiene que compartir información entre funciones, las variables se pasan como parámetros.
d. **Porque su ámbito es el local de la función.** Falso. Las variables globales son accesibles desde cualquier función

10. Dado el siguiente programa, señale la afirmación correcta :

```
#include <stdio.h>
int Intercambio(int*, int*);

int main(void){
   int x,y,z;
   x=5;
   y=10;
   z=Intercambio(&x,&y);
   printf("x=%i\t y=%i\n", x, y);
   system("pause");
   return 0;
}

int Intercambio(int* a, int* b)
{
      int aux;
      aux=b;
      b=a;
      a=aux;
      return(0);
}
```

Solución: Aunque los parámetros de la función Intercambio se pasan por referencia, el código contenido en el cuerpo de esta función no modifica el valor de los parámetros pasados. Para lograr este objetivo deberíamos usar `*a` y `*b` en lugar de `a` y `b`. Por tanto, tras la llamada a la función los valores de `x` e `y` siguen siendo 5 y 10 respectivamente.

a. **Imprime por pantalla x=5 y=10@**
b. **Imprime por pantalla x=10 y=5**
c. **El programa no compila porque hay un error en la declaración de la función (prototipo).**
d. **El programa no compila porque hay un error en el cuerpo de la función.**