

Revisar entrega de examen: UF1 2a Convocatoria

Usuario	
Curso	2209_ASIR_MP03_Programación básica
Examen	UF1 2a Convocatoria
Iniciado	4/02/23 19:00
Enviado	4/02/23 20:14
Fecha de vencimiento	4/02/23 20:30
Estado	Completado
Puntuación del intento	7,5 de 10 puntos
Tiempo transcurrido	1 hora, 14 minutos
Resultados mostrados	Todas las respuestas, Respuestas enviadas, Respuestas correctas, Comentarios, Preguntas respondidas incorrectamente

Pregunta 1

2,5 de 2,5 puntos



Explica cuándo usarías y qué ventajas o desventajas tienen las siguientes instrucciones cuando se quiere recoger por teclado una cadena o array de caracteres.

Una vez explicado, pon un ejemplo de uso de almacenar una cadena en la siguiente variable: `char respuesta[20];`

SCANF: con %s

GETS:

FGETS:

Respuesta seleccionada: -Las tres funciones sirven para leer información por teclado.

scanf no permite desdeñar espacios en blanco dentro de la cadena.
gets permite guardar espacios en blanco pero no está protegida contra buffer overflow.
fgets es la mejor opción ya que está protegida contra el buffer overflow.

ejemplo:

```
char respuesta[20];  
printf("Respuesta ");  
scanf("%s", respuesta);
```

ejemplo:

```
char respuesta[20];  
printf("Respuesta ");  
gets(respuesta);
```

ejemplo:

```
char respuesta[20];  
printf("Respuesta ");  
fgets(cadena,20,stdin);
```

Respuesta correcta: [None]

Comentarios para respuesta: [No se ha dado ninguna]

Pregunta 2

5 de 5 puntos



Queremos hacer un programa en C que pueda almacenar en un array de cadenas el nombre de varios clientes, siendo 100 el tamaño máximo del array.

El programa preguntará al usuario cuántos clientes quiere introducir. Este valor debe estar validado dentro de lo permitido en el array.

Una vez sabemos cuántos clientes se van a introducir, se pedirá al usuario que vaya introduciendo cada nombre uno a uno. Posteriormente, se deben mostrar los nombres por pantalla.

Marca y corrige los errores que encuentres en el programa. No hace falta que vuelvas a copiar el código.

Puedes indicar:

ERROR 1: (blabla) **SOLUCIÓN 1:** (blabla)

ERROR 2: (blabla) **SOLUCIÓN 2:** (blabla)

etc.

```
#include <stdio.h>

#define MAXCLIENTES 100
#define MAXCADENA 20

int main(){

    char vClientes[MAXCLIENTES];
    int numE;

    printf("Introduce cuantos nombres de clientes quieres introducir:");
    scanf("%d",&numE);

    //Comprobamos que esté dentro del rango permitido
    while(numE<1 || numE>MAXCADENA)
    {
        printf("\nNumero incorrecto.");
        printf("\nIntroduce un numero entre 1 y %d:", MAXCLIENTES);
        fflush(stdin);
        scanf("%d",&numE);
    }

    //Pedimos al usuario que introduzca los nombres de los clientes uno a uno
    for(int i=1; i<=numE; i++)
    {
        printf("Introduce el nombre: %d: ", i+1);
        fflush(stdin);
        fgets(vClientes, MAXCLIENTES, stdin);
    }

    //Muestra por pantalla los nombres de los clientes que hemos introducido en el array
    printf("\nLISTADO DE CLIENTES ");
    for(int i=0; i<numE; i++)
    {
        printf("\nCliente %d: %d", i+1, vClientes[i]);
    }
    printf("\nTotal Clientes: %d", vClientes[numE-1]);

    system("pause");
    return 0;
}
```

Respuesta seleccionada:	ERROR 1 ==> char vClientes[MAXCLIENTES]; SOLUCIÓN ==> char vClientes[MAXCLIENTES] [MAXCADENA]; ERROR 2 ==> while(numE<1 numE>MAXCADENA) SOLUCIÓN ==> while(numE<1 numE>MAXCLIENTES) ERROR 3 ==> for(int i=1; i<=numE; i++) SOLUCIÓN ==> for(int j=0; j<numE; j++) ERROR 4 ==> fgets(vClientes, MAXCLIENTES, stdin); SOLUCIÓN ==> fgets(vClientes[i], MAXCADENA, stdin); ERROR 5 ==> printf("\nCliente %d: %d", i+1, vClientes[i]); SOLUCIÓN ==> printf("\nCliente %d: %s", i+1, vClientes[i]); ERROR 6 ==> printf("\nTotal Clientes: %d", vClientes[numE-1]); SOLUCIÓN ==> printf("\nTotal Clientes: %d", numE);
Respuesta correcta:	[None]
Comentarios para respuesta:	[No se ha dado ninguna]

Pregunta 3

0 de 2,5 puntos



Queremos hacer un programa en C que solicite un número natural positivo (numLimite) de 3 cifras al usuario . Una vez introducido, el programa irá pidiendo al usuario números de 2 cifras hasta que, o bien la suma de éstos sea superior al número 'numLimite' pedido, o se hayan pedido más de 10 números de 2 cifras.

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

int main()
{
    SetConsoleOutputCP(CP_UTF8);
    int numLimite;
    int num;
    int acum[1]; //Para acumular la suma de los numeros introducidos por el usuario
    int cont [2]; //Numero de intentos que ha utilizado el usuario










    printf("Introduce un número de 3 cifras positivo que será el límite:");
    scanf("%d", &numLimite);
    //Comprobar que el número límite sea de 3 cifras
    while ([3])
    {
        printf("\nEl numero debe ser de 3 cifras.");
        printf("\nIntroduce un número de 3 cifras que será el límite:");
        scanf("%d", &numLimite);
    }
    //Pide numeros al usuario continuamente hasta que se cumpla la condicion de salir
    do
    {
        printf("Introduce un número de dos cifras:");
        scanf("%d", &num);










        //Comprueba que el numero introducido es de dos cifras.
        while ([4])
        {
            printf("\nEl número debe ser de 2 cifras.");
            printf("\nIntroduce un número de dos cifras:");
            [5] //Vuelve a pedir que se introduzca el numero
        }

        acum = [6]; //Acumulamos el numero introducido
        [7] //Incrementamos en 1 el numero de intentos
        printf("Acum: %d \t Cont: %d \n", acum, cont);

    } while ([8]); //Deja de pedir numeros cuando la suma de todos los introducidos sea superior o igual a numLimite, o si
    superamos 10 intentos.
    //Comprobamos si se ha superado el numero limite o no
    if ([9])
    {
        printf("\nLa suma de los números introducidos es mayor o igual a %d\n", numLimite);
    }
    else
    {
        printf("\nLa suma de los números introducidos es menor a %d\n", numLimite);
    }

    system("pause");
    return (0);
}
```

Respuesta especificada para: 1  numLimite
 Respuesta especificada para: 2  num
 Respuesta especificada para: 3  numlimite< || num
 Respuesta especificada para: 4  numlimite <|| num
 Respuesta especificada para: 5  printf:"vuelve a introducir un numeor"
 Respuesta especificada para: 6  numlimite == num
 Respuesta especificada para: 7  Acum 1++
 Respuesta especificada para: 8  [No se ha dado ninguna]
 Respuesta especificada para: 9  int i = num

Respuestas correctas para: 1		
Método de evaluación	Respuesta correcta	Distingue entre mayúsculas y minúsculas
 Correspondencia exacta	=0	
Respuestas correctas para: 2		
Método de evaluación	Respuesta correcta	Distingue entre mayúsculas y minúsculas
 Correspondencia exacta	=0	
Respuestas correctas para: 3		
Método de evaluación	Respuesta correcta	Distingue entre mayúsculas y minúsculas
 Correspondencia exacta	numLimite <= 100 numLimite > 1000	
Respuestas correctas para: 4		
Método de evaluación	Respuesta correcta	Distingue entre mayúsculas y minúsculas
 Correspondencia exacta	num < 10 num > 100	
Respuestas correctas para: 5		
Método de evaluación	Respuesta correcta	Distingue entre mayúsculas y minúsculas
 Correspondencia exacta	scanf("%d", &num);	
Respuestas correctas para: 6		
Método de evaluación	Respuesta correcta	Distingue entre mayúsculas y minúsculas
 Correspondencia exacta	acum + num	
Respuestas correctas para: 7		
Método de evaluación	Respuesta correcta	Distingue entre mayúsculas y minúsculas
 Correspondencia exacta	cont++;	
Respuestas correctas para: 8		
Método de evaluación	Respuesta correcta	Distingue entre mayúsculas y minúsculas
 Correspondencia exacta	cont < 10 && acum < numLimite	
Respuestas correctas para: 9		
Método de evaluación	Respuesta correcta	Distingue entre mayúsculas y minúsculas
 Correspondencia exacta	acum >= numLimite	