Pregunta 1 3 de 3 puntos

Escribir una función llamada `tri` que imprime en pantalla un triángulo rectángulo. Lo hace ubicando el ángulo recto arriba a la derecha, el interior debe estar vacío y el borde debe tener asteriscos. La función recibe el parámetro `n` (longitud de los lados) y no devuelve nada. Debe funcionar para todo $1 \leq n \leq 60$ (se debe validar). No se permite utilizar arreglos ni matrices.

Por ejemplo:
`tri(4)` imprime:

```
****
 * *
  **
   *
```

`tri(7)` imprime:

```
*******
 *    *
  *   *
   *  *
    * *
     **
      *
```

`tri(1)` imprime:

```
*
```

Respuesta seleccionada:

```
void tri (unsigned int n)    //recibo un entero positivo
{
    int i, j;
    //verifico que el n que se le pasa como parametro sea valido, o sea que este entre 1 y 60
    if ((n<1) || (n>60))
    {
        printf("error, superaste los limites de n\n");
    }
    else
    {
        for(i=0; i<n; i++)    //voy recorriendo por fila
        {
            for(j=0; j<n; j++)    //voy recorriendo por columna
            {
                if((i==j) || (j==n-1) || (i==0))    //las condiciones en el cual se imprime el *
                {
                    printf("*");
                }
                else
                {
                    printf(" ");
                }
            }
            printf("\n");    //hago el salto de linea para ir a la siguiente fila
        }
    }
}
```

Comentarios para respuesta: Valida OK; comenta OK

Compila OK-: falta incluir <stdio.h>

Funciona OK en todos los casos!

Pregunta 2 1 de 3 puntos

Escribir una función que determine si un *string* está presente dentro de otro *string* y devuelva el resultado.
Por ejemplo:
- `"u"` está presente en `"Mundo"` (debe devolver 1)
- `"ca"` está presente en `"Me pica la cabeza"` (debe devolver 1)
- `"la capa"` no está presente `"Me pica la cabeza"` (debe devolver 0)
- `"782"` no está presente `"12345678"` (debe devolver 0)

Respuesta seleccionada:

```
int is_in(char* str1, char* str2)  //str1 es el string con el que quiero ver si esta en str2
{
    int i, j, flag=0;
    int n1 = sizeof(str1);   //averiguo el tamaño de los strings
    int n2 = sizeof(str2);
    for(i=0; i<(n2-1); i++)  //recorro hasta n2-1, teniendo en cuenta el terminador \0
    {
        if(str1[j]==str2[i])  //comparo si la primera letra es igual o no
        {
            for(j=0; j<(n1-1); j++)  //si es igual, sigo comparando el resto de la palabra
            {
                if((i+j)>=n2)    //si me paso del 2 string, salgo del for
                {
                    break;
                }
                if(str1[j]==str2[i+j])
                {
                    if(j==n1-2)   //verifico si ya termine de comparar, pero no comparo el terminador del str1
                    {
                        flag = 1;    //seteo el flag en 1
                    }
                    continue;
                }
                break;               //si no hay coincidencia, salgo instantaneamente
            }
        }
        if(flag==1)
        {
            break;
        }
    }
    return flag;
}
```

Comentarios para respuesta: Comenta OK

Compila OK

NO funciona!

- Error de concepto: calcula tamaño de string mediante sizeof! -> -1

- Nunca inicializa j, programa genera segmentation fault -> -0,5

- Uso desprolijo y evitable de break y continue -> -0,5

Corrigiendo esto funciona OK

Pregunta 3

2 de 2 puntos

Indicar que imprime en consola el siguiente programa. **Justificar la respuesta**.

```
#include <stdio.h>

#define D   10
#define X   20

char d = 30;
char x = 40;

int main(void)
{
    char d = 50;

    printf("D x %d %X\n", X, d);

    x = 0x99;
    d = x == 0x99;
    printf("%d %x\n", x, d);

    return 0;
}
```

Respuesta seleccionada:

Defino como constantes:
D = 10
X = 20
Defino como variables globales:
char d = 30;
char x = 40;
Dentro del main declaro y defino nuevamente una variable local:
char d = 50;
La siguiente linea (printf("D x %d %X\n", X, d);) imprime:
D x 20 32
Primero imprime "D x", luego con el especificador de formato %d, toma el valor de X y lo imprime como un tipo de dato int. Por lo que imprime el valor 20 del #define. Luego le sigue el especificador %X, el cual indica que imprime el valor entero en formato hexa. Entonces toma el valor d=50 y lo convierte en hexa, el cual vale 32. Y por ultimo hace un salto de linea.

Luego declara y define nuevamente la variable x=0x99; y d=x==0x99;
Primero le doy el valor 0x99 (hexa) a la variable x. Luego a la variable "d" le doy el valor de un booleano. Porque lo que hago en "x==0x99" es comparar si el valor que tiene "x" es igual a "0x99".
Luego en la siguiente linea (printf("%d %x\n", x, d);) imprime:
-103 0
Los tipos de variable char tiene un rango de -128 a 127 decimal. Como le cambie el valor a la variable "x" de tipo char y le asigne el valor de 0x99 = 153 en decimal. Tengo un overflow y me pase a la parte negativa. Por lo tanto como el maximo es 127 y tengo el valor 153, hago 153-127=26. Al valor 26 se lo sumo al -128, entonces -128+26=-102. Sin embargo, teniendo en cuenta el 0, el valor final seria -103 en formato decimal (int).
Luego en la parte donde imprime la variable "d" en formato hexa, debido a que la variable "x" por overflow, ya no tiene el valor de 0x99=153, sino -103. Comparando 153 y -103 me devuelve "falso" o sea un "0". Por lo tanto en la variable "d" se guarda un 0 y se imprime el 0 en formato hexa, que es 0.

Comentarios para respuesta: Todo OK!

Pregunta 4

2 de 2 puntos

1) En una terminal de Linux se desea:
a. Compilar un programa guardado en el archivo `main.c`, mostrando todos los warnings, definiendo la constante `DEBUG`, creando el archivo ejecutable `recuperatorio` y generando la información para poder "debuggearlo".
b. Ejecutar el archivo compilado.
c. "Debuggear" el programa.
Escribir los comandos para realizar dichas acciones.

2) Nombrar al menos 2 comandos de *gdb*, indicando para que sirven.

Respuesta seleccionada:

```
1)
a.
gcc main.c -Wall -D DEBUG -o recuperatorio -g
b.
./recuperatorio
c.
gdb recuperatorio
2)
comando 1: breakpoint 10 o b 10
El comando de breakpoint sirve para colocar un punto de corte en la ejecucion del programa. Luego con el comando "run" el programa se ejecuta hasta la linea 10 del programa.
comando 2: n o next
El comando de next sirve para ir ejecutando la siguiente linea de comando durante el debug
```

Comentarios para respuesta: OK