	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	M. I. Marco Antonio Martínez Quintana
Asignatura:	Estructura de Datos y Algoritmos I
Grupo:	15
No de Práctica(s):	1
Integrante(s):	Sánchez Hernández Marco Antonio
No. de Equipo de cómputo empleado:	No aplica
No. de Lista o Brigada:	
Semestre:	2021-2
Fecha de entrega:	15 de marzo del 2021
Observaciones:	La práctica se realizó en el sistema operativo Arch Linux, el cual cuenta con una distribución de teclado en inglés US. Se utilizó Unicode para imprimir las vocales con acentos y algunos caracteres especiales.

CALIFICACIÓN: _____

Sistema de Gestión de Calidad

Objetivo:

Utilizar arreglos unidimensionales y multidimensionales para dar solución a problemas computacionales.

Introducción

"*Un arreglo es un conjunto de datos contiguos del mismo tipo con un tamaño fijo definido al momento de crearse*" (Solano J., 2019). Un arreglo puede ser unidimensional, los datos son ordenados en un solo nivel, o multidimensionales, los datos pueden ser ordenados en varios niveles. Los datos contenidos en un arreglo unidimensional son almacenados en memoria de la misma forma que los arreglos multidimensionales.

A cada dato, que está contenido en un arreglo, se le asigna un índice, el cual indica su lugar en el arreglo. En la mayoría de lenguajes de programación, como en el caso del lenguaje C, al primer elemento le corresponde el índice cero, mientras que al último le corresponde el índice n-1. Si se trata de arreglos multidimensionales, se deben utilizar múltiples índices para referirse a un elemento, ejemplo, si se tiene un arreglo tridimensional, se utilizarán tres índices para denotar un elemento, cada uno corresponde a la posición que este ocupa en cada una de las tres dimensiones. Análogamente, el índice de un arreglo puede ser visto como las coordenadas de un plano, donde el número de dimensiones, será el número de componentes.

Los arreglos pueden ser de dos tipos, contiguos, no pueden ser redimensionados durante el tiempo de ejecución, y ligados, pueden ser redimensionados durante el tiempo de ejecución.

Ejemplo 1: escitala espartana

Para la realización de este programa, se tomó como base el código presente en la práctica número 1 del "*Manual de prácticas del laboratorio de Estructuras de datos y algoritmos I*".

Para mejorar el rendimiento del programa y dar mayor claridad y orden al código fuente, el programa fue dividido en dos archivos. En el primer archivo, llamado "*main.c*", se encuentra la función main, mientras que, el archivo "*mensaje.h*" es una librería que contiene las funciones cifrar y descifrar.

Enlace al repositorio de GitHub que contiene el código fuente del programa: https://github.com/marco-sanchez-est/EDAI/tree/main/codigos/escitalaEspartana

Archivo “*main.c*”

```c
/*
 * Autor: Marco Antonio Sanchez Hernandez
 * Nacionalidad: Mexicana
 * Fecha de elaboracion: 11-03-2021
 * Ultima modificacion: 14-03-2021
 * Sistema Operativo utilizado: Arch Linux
 * Kernel: Linux 5.11.5-arch1-1
 *
*/

#include <stdio.h>
#include <stdlib.h> //Libreria estandar, contiene la funcion int system(const char*string).
#include <string.h> //Liberia string, contiene la funcion strcpy que permite limpiar un arreglo.
#include "mensaje.h" //Libreria creada, contiene las funciones para cifrar y descifrar un mensaje.

int main() {

  //Variables.
  unsigned short opcion = 0;
  int ren, col; //Almacenan el tamanio de la escitala.
  char mensaje[ren*col]; //Almacena el mensaje que se desea cifrar o descifrar.

  do{
    system("clear"); //Limpia la pantalla ejecutando el script clear propio de la terminal.
    opcion = ren = col = 0; //Inicia el valor de las variables en cero.
    strcpy(mensaje,"\0"); //Limpia el arreglo mensaje.

    //Menu de opciones.
    printf("\n\t*** ES\u00CDTALA ESPARTANA ***\n");
    printf("\u00BFQu\u00E9 desea realizar?\n");
    printf("1) Crear mensaje cifrado.\n");
    printf("2) Descifrar mensaje.\n");
    printf("3) Salir.\n\n");
    scanf("%hu", &opcion); //Leer opcion ingresada.

    switch(opcion) {
      case 1:
        printf("\nIngrese el tama\u00F1o de la escitala:\n");
        printf("\nRenglones: ");
        scanf("%d",&ren);

        //No permite ingresar valores negativos.
        if (ren < 1) {
          printf("El valor ingresado no es v\u00E1lido\n");
```

```c
          break;
        }

        printf("Columnas: ");
        scanf("%d",&col);

        //No permite ingresar valores negativos.
        if (col < 1) {
          printf("El valor ingresado no es v\u00E1lido\n");
          break;
        }

        printf("\nEscriba el texto a cifrar:\n");
        scanf("%s",mensaje);
        cifrar(ren,col,mensaje); //Pasa los argumentos ren, col y mensaje a la funcion cifrar
        system("read -n 1 -s -p\"\nPresione cualquier tecla para continuar\""); //Espera a que el usuario presione alguna tecla para reanudar la ejecucion.
      break;

      case 2:
        printf("\nIngrese el tama\u00F1o de la escitala:\n");
        printf("\nRenglones: ");
        scanf("%d",&ren);

        //No permite ingresar valores negativos.
        if (ren < 1) {
          printf("El valor ingresado no es v\u00E1lido\n");
          break;
        }

        printf("Columnas: ");
        scanf("%d",&col);

        //No permite ingresar valores negativos.
        if (col < 1) {
          printf("El valor ingresado no es v\u00E1lido\n");
          break;
        }

        printf("\nEscriba el texto a descifrar:\n");
        scanf("%s",mensaje);
        descifrar(ren,col,mensaje); //Pasa los argumentos ren, col y mensaje a la funcion descifrar
        system("read -n 1 -s -p\"\nPresione cualquier tecla para continuar\""); //Espera a que el usuario presione alguna tecla para reanudar la ejecucion.
```

```c
      break;

    }
  }while(opcion != 3);

  return 0;

}
```

Librería “*mensaje.h*”

```c
/*
 * Autor: Marco Antonio Sanchez Hernandez
 * Nacionalidad: Mexicana
 * Fecha de elaboracion: 11-03-2021
 * Ultima modificacion: 14-03-2021
 * Sistema Operativo utilizado: Arch Linux
 * Kernel: Linux 5.11.5-arch1-1
 *
*/

/* Definicion de la funcion cifrar, recibe como parametros:
 * Dos valores enteros.
 * Un arreglo de caracteres que tenga por tamanio el producto de los valores enteros.
 * No retorna valor alguno. */
void cifrar(int ren, int col, char texto[ren*col]) {

  // Variables locales.
  int i, j, k = 0; // Variables de iteracion.
  char str[ren][col]; /* Arreglo bidimensional de caracteres, tiene por tamanio los
                         valores enteros que recibe como parametro la funcion*/

  //Ciclo para convertir un arreglo unidimensional en uno bidimensional.
  for (i = 0; i < ren ; i++)
    for (j = 0; j < col; j++)
      str[i][j] = texto[k++];
    //Almacena el valor contenido en el indice k, en el indice i,j del arreglo bidimensional.

  printf("\nEl texto en la tira queda de la siguiente manera:\n");

  //Ciclo para mostrar en pantalla la cadena de caracteres cifrada.
  //Imprime caracter a caracter.
  for (i = 0; i < col; i++)
    for (j = 0; j < ren; j++)
      printf("%c",str[j][i]);

  printf("\n");

}

/* Definicion de la funcion descifrar, recibe como parametros:
 * Dos valores enteros.
 * Un arreglo de caracteres que tenga por tamanio el producto de los valores enteros.
 * No retorna valor alguno. */
void descifrar(int ren, int col, char texto[ren*col]) {
```

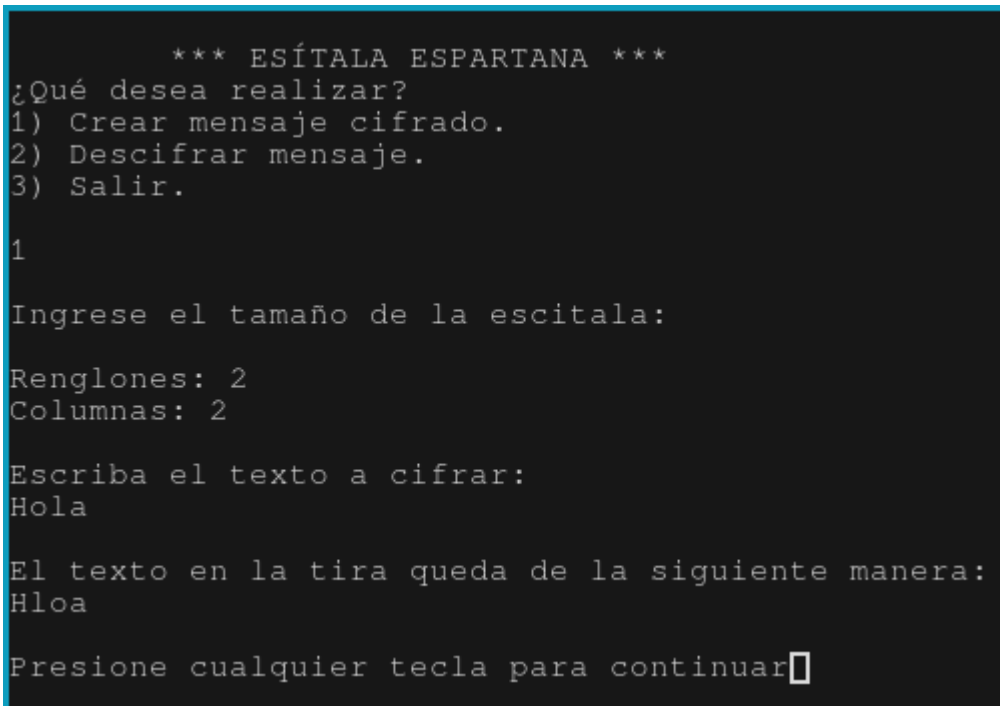
mensaje.h 1,1 Top

```c

  //Variables.
  int i, j, k = 0; // Variables de iteracion.
  char str[ren][col]; /* Arreglo bidimensional de caracteres, tiene por tamanio los
                         valores enteros que recibe como parametro la funcion.*/

  //Ciclo para convertir un arreglo unidimensional en uno bidimensional.
  for (i = 0; i < col; i++)
    for (j = 0; j < ren; j++)
      str[j][i] = texto[k++];
    //Almacena el valor contenido en el indice k, en el indice j,i del arreglo bidimensional.

  printf("\nEl texto descifrado es:\n");

  //Ciclo para mostrar en pantalla la cadena de caracteres descifrada.
  //Imprime caracter a caracter.
  for (i = 0; i < ren; i++)
    for (j = 0; j < col; j++)
      printf("%c",str[i][j]);

  printf("\n");

}
```

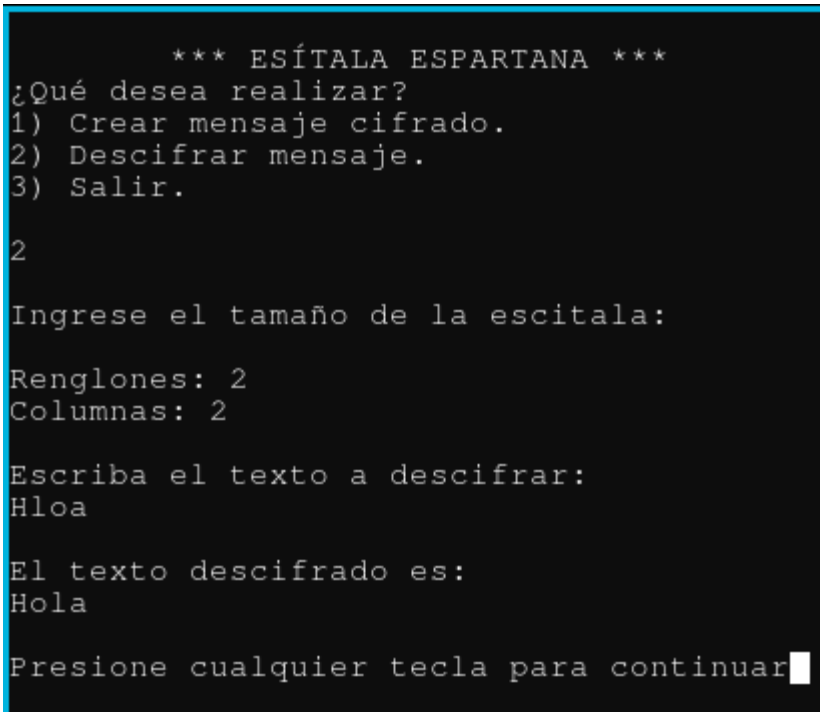
mensaje.h 67,1 Bot

Ejecución del programa

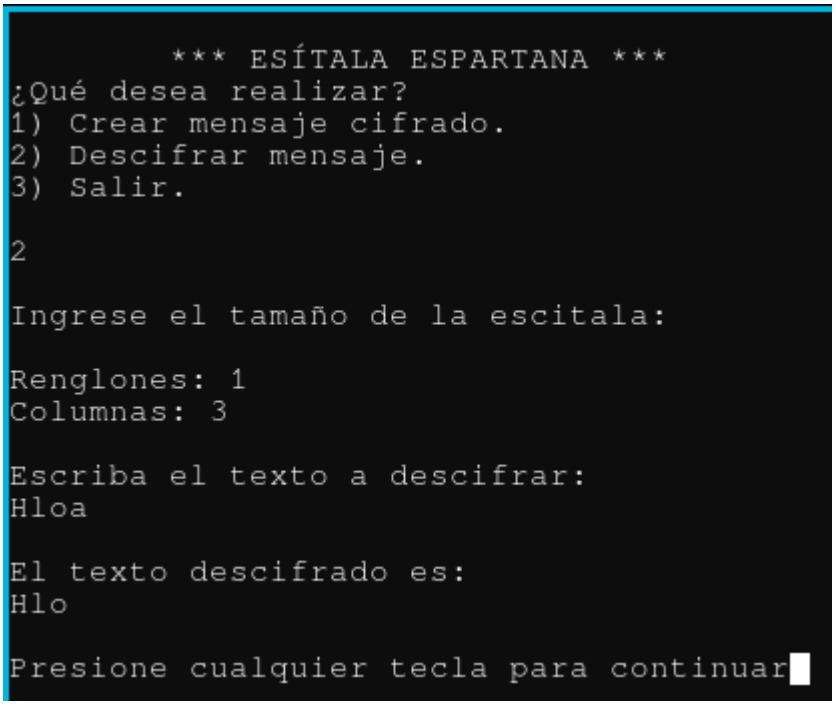
En esta captura se muestra la creación de una escitala de dos renglones por dos columnas mediante el uso de la opción número uno, que se utiliza para cifrar la cadena de caracteres “Hola”, dando como resultado la cadena “Hloa”.

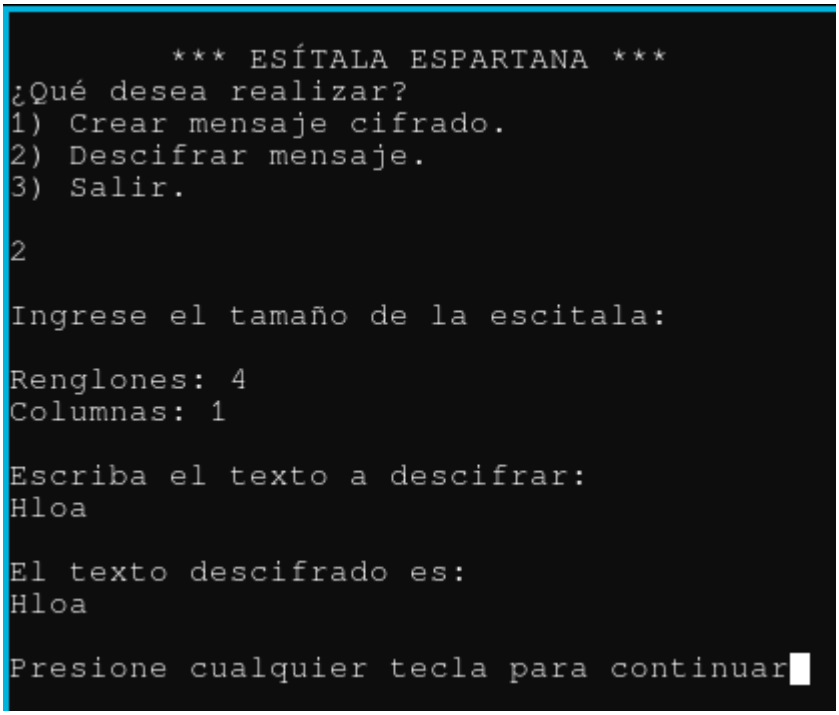


En esta captura se muestra la creación de una escitala de dos renglones por dos columnas mediante el uso de la opción número dos, la cual es utilizada para descifrar la cadena de caracteres “Hloa”, dando como resultado la cadena “Hola”.

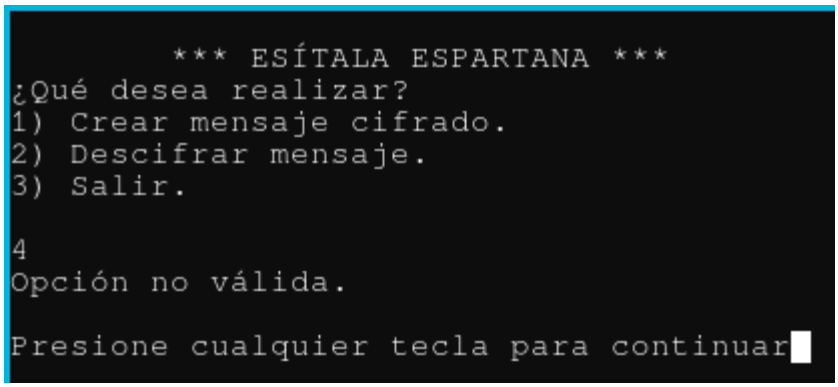


Si el usuario utiliza una escitala que no tenga el mismo número de renglones y columnas que la utilizada para crear el mensaje, el mensaje no será descifrado correctamente, dando como resultado una nueva cadena ilegible.

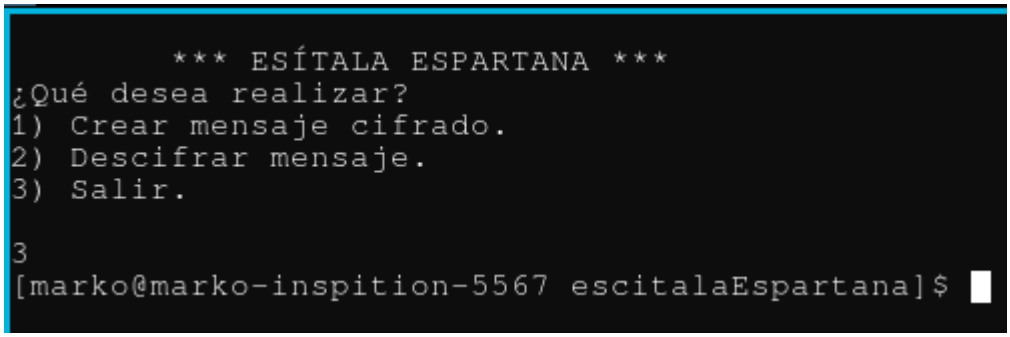




Si el usuario ingresa en el menú algún valor diferente de uno, dos o tres, el programa le notificará que ha ingresado una opción no válida.



Si el usuario ingresa la opción número tres, el programa terminará la ejecución.



Ejercicio 1: Sudoku

Enlace al repositorio de GitHub que contiene el código fuente del programa: https://github.com/marco-sanchez-est/EDAI/tree/main/codigos/sudoku

Archivo “*main.c*”

```c
/*
 * Autor: Marco Antonio Sanchez Hernandez
 * Nacionalidad: Mexicana
 * Fecha de elaboracion: 13-03-2021
 * Ultima modificacion: 14-03-2021
 * Sistema Operativo utilizado: Arch Linux
 * Kernel: Linux 5.11.5-arch1-1
*/

#include <stdio.h>
#include <stdlib.h> //Libreria estandar, contiene la funcion system (const char*string).
#include "show.h" //Liberiria creada, contiene la funcion que muestra en pantalla y actualiza el sudoku.

int main(){

  //Arreglo bidimensional que contiene el sudoku resuelto correctamente.
  int sudoku[9][9] = {4,2,7,  1,3,9,  5,6,8,  9,1,5,  8,7,6,  3,4,2,  6,8,3,  5,4,2,  1,9,7,  2,5,6,  9,1,8,  4,7,3,  3,4,9,  2,6,7,  8,5,1,  8,7,1,  4,5,3,  9,2,6,  5,9,8,  6,2,1,  7,3,4,  7,6,4,  3,8,5,  2,1,9,  1,3,2,  7,9,4,  6,8,5};

  //Arreglo bidimensional que contiene las respuestas ingresadas por el usuario.
  int respuestas[9][9] = {4,2,7,  1,0,0,  0,6,8,  0,0,5,  0,0,6,  3,0,0,  6,0,3,  0,0,0,  1,0,0,  2,0,0,  0,1,0,  4,0,0,  3,4,0,  0,6,7,  0,5,1,  8,0,1,  0,5,0,  0,2,0,  0,9,0,  0,0,0,  7,3,0,  7,0,4,  3,0,0,  2,0,9,  0,3,2,  0,9,4,  6,0,0};

  //Variables
  int x, y, z, a = 0, errores = 0, opcion = 0;

  do{
    //Muestra las instrucciones de funcionamiento al usuario.
    printf("\n\t\t*** Sudoku ***\n\n");
    printf("Ingrese tres n\u00FAmeros enteros separados por una coma\n");
    printf("-Deben ser numeros entre 1 y 9, incluyendo el 1 y 9\n");
    printf("-El primer n\u00FAmero corresponde al rengl\u00F3n\n");
    printf("-El segundo n\u00FAmero corresponde a la columna\n");
    printf("-El tercer n\u00FAmero corresponde a su respuesta\n\n");
    mostrar(respuestas); //Muestra en pantalla el sudoku que contiene las respuestas que da el usuario.
    printf("\n\nErrores: %d",errores); //Muestra el numero de errores cometidos.
    printf("\nIngrese su respuesta: ");
    scanf("%d, %d, %d", &x, &y, &z); //Lee de la entrada estandar 3 valores separados por una coma y los almacena en las variables x, y, z.

    //Condicion para limitar que los valores enteros ingresados sean mayores que cero, y menores que 10.
    if (x < 1 || x > 9 || y < 1 || y > 9 || z < 1 || z > 9) {
      printf("Los valores ingresados no v\u00E1lidos\n");
      system("read -n 1 -s -p\"Presione cualquier tecla para continuar\""); //Espera a que el usuario presione alguna tecla para c@@@
main.c                                                                                                    1,1           Top
```

```c
    }

    else{
      //Compara si el valor ingresado del usuario es igual al contenido en el arreglo que contiene las respuestas corresctas en la posicion [x][y].
      if (z == sudoku[x-1][y-1]) {
        //Asigna el valor z ingresado por el usuario a la posicion [x-1][y-1] del arreglo que contiene las respuestas del usuario.
        respuestas[x-1][y-1] = z;
        a = 0;

        //Ciclo que realiza la suma de todos los elementos contenidos en el arreglo respuests.
        for (int i = 0; i < 9; i++)
          for (int j = 0; j < 9; j++)
            a = a + respuestas[i][j];
      }

      else {
        //Notifica al usuario que cometio un error y espera a que este presione una tecla.
        printf("Error:\n");
        system("read -n 1 -s -p\"Presione cualquier tecla para continuar\""); //Espera a que el usuario presione alguna tecla para reanudar la ejecucion.

        //Aumenta el numero de errores.
        errores = errores +1;

        //Si el usuario comete 5 erroes, el programa termina.
        if (errores == 5) {
          //Asigna el valor 2 a la variable opcion para mostrar un mensaje de derrota.
          opcion = 2;
          break;
        }
      }
    }

    system("clear"); //Limpia la pantalla ejecutando un script propio de la terminal de comandos.
  }while( a != 405); //El ciclo se repite hasta que la suma de los elementos del arreglo respuestas sea 405.

  //Permite seleccionar el mensaje final a mostrar.
  switch(opcion) {
    //Este mensaje se muestra si el usuario completa el sudoku correctamente.
    default:
      printf("Enhorabuena, lo lograste\n");
      printf("No. de errores: %d\n",errores);
    break;
main.c                                                                                                    82,34         77%
"main.c" 95L, 4177C written
```

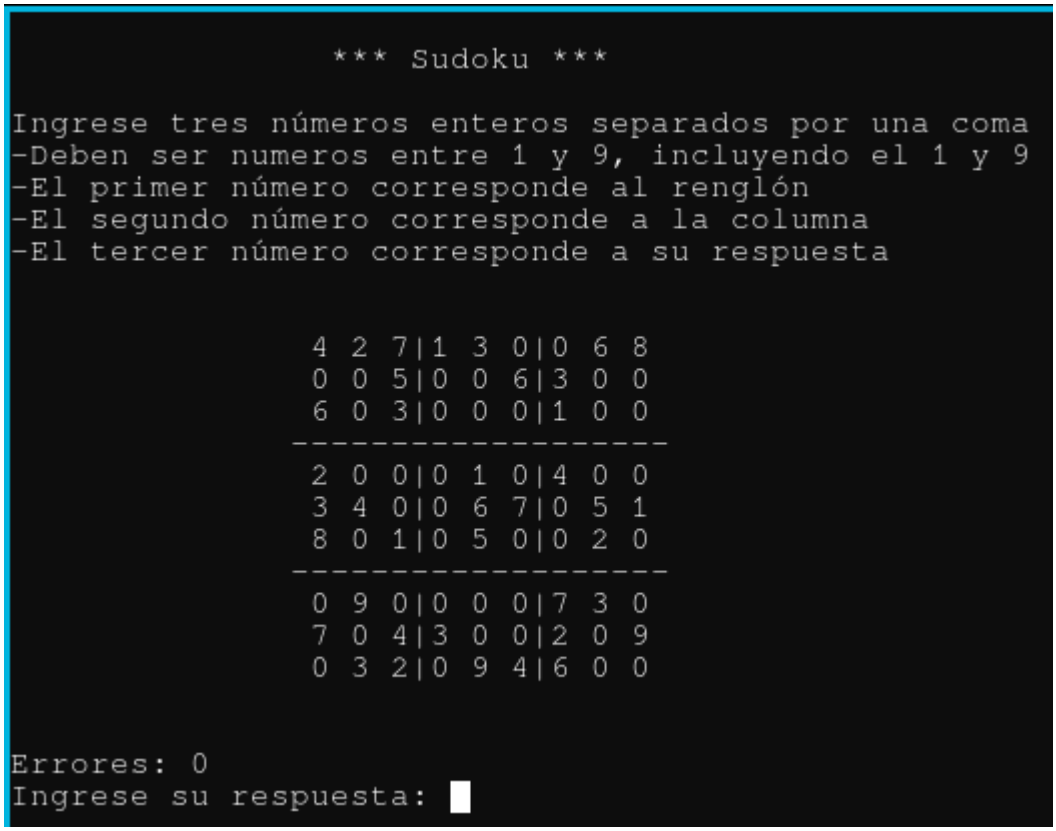
```
    //Este mensaje se muestra si el usuario cometio 5 errores.
    case 2:
      system("clear"); //Limpia la pantalla ejectunado un script propio de la terminal.
      printf("Intento terminado\n");
      printf("Suerte para la pr\u00F3xima\n\n");
    break;
  }

  return 0;

}
main.c                                                                95,1          Bot
```

Librería “*show.h*”

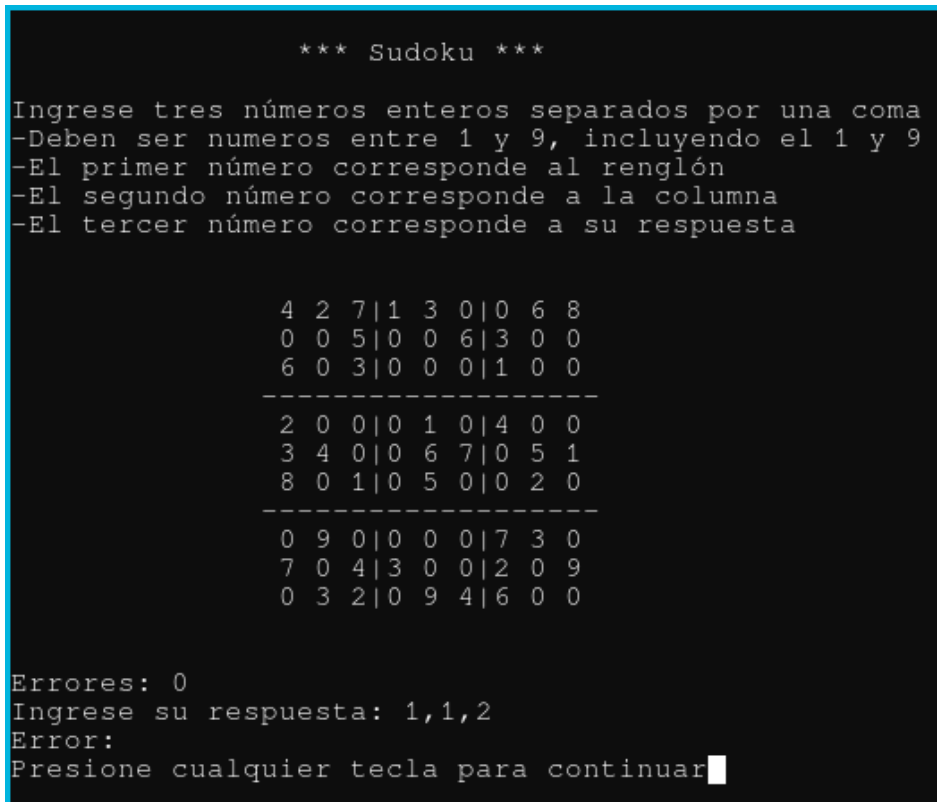
```
/*
 * Autor: Marco Antonio Sanchez Hernandez
 * Nacionalidad: Mexicana
 * Fecha de elaboracion: 13-03-2021
 * Ultima modificacion: 14-03-2021
 * Sistema Operativo utilizado: Arch Linux
 * Kernel: Linux 5.11.5-arch1-1
 */

/*Definicion de la funcion mostrar
-Recibe como parametro un arreglo bidimensional de 9x9 de numeros enteros.
-No regresa ningun valor.
 */
void mostrar(int arreglo[9][9]) {

  //Variables de iteracion
  int i, j;

  /*Bloque de codigo que permite imprimir el sudoku con el formato deseado*/
  printf("\n");
  for (i = 0; i < 9; i++) {
    //Imprime una linea punteada cada tres lineas para separar los renglones.
    if (i == 3 || i ==6)
      printf("\t      -------------------\n");
    printf("\t      "); //Inserta una tabulacion y 6 espacios antes de imprimir los numeros.
    for (j = 0; j < 9; j++) {
      //Imprime una barra vertical cada tres numeros.
      if (j == 3 || j == 6)
        printf("|%d",arreglo[i][j]);
      else {
        printf(" %d",arreglo[i][j]);
      }
    }
    printf("\n");
  }

}
```

Las siguientes dos capturas muestran al usuario queriendo ingresar el valor “3”, en el renglón “1” y la columna “5”, al ser esta una respuesta correcta, el sudoku se actualiza y ahora muestra el valor “3” en la posición indicada.

```
[marko@marko-inspition-5567 sudoku]$ ./sudoku

                *** Sudoku ***

Ingrese tres números enteros separados por una coma
-Deben ser numeros entre 1 y 9, incluyendo el 1 y 9
-El primer número corresponde al renglón
-El segundo número corresponde a la columna
-El tercer número corresponde a su respuesta


              4 2 7|1 0 0|0 6 8
              0 0 5|0 0 6|3 0 0
              6 0 3|0 0 0|1 0 0
             -------------------
              2 0 0|0 1 0|4 0 0
              3 4 0|0 6 7|0 5 1
              8 0 1|0 5 0|0 2 0
             -------------------
              0 9 0|0 0 0|7 3 0
              7 0 4|3 0 0|2 0 9
              0 3 2|0 9 4|6 0 0


Errores: 0
Ingrese su respuesta: 1,5,3
```



Las siguientes dos capturas muestra lo que ocurre cuando el usuario no ingresa un valor correcto en la posición indicada. En la primera captura se muestra un mensaje que indica error y se espera a que el usuario presione alguna tecla, la segunda captura muestra el mismo sudoku que se tenía en la primera captura, y aumenta en uno el indicador de errores que ha cometido el usuario.



```
                *** Sudoku ***

Ingrese tres números enteros separados por una coma
-Deben ser numeros entre 1 y 9, incluyendo el 1 y 9
-El primer número corresponde al renglón
-El segundo número corresponde a la columna
-El tercer número corresponde a su respuesta


                 4 2 7|1 3 0|0 6 8
                 0 0 5|0 0 6|3 0 0
                 6 0 3|0 0 0|1 0 0
                -------------------
                 2 0 0|0 1 0|4 0 0
                 3 4 0|0 6 7|0 5 1
                 8 0 1|0 5 0|0 2 0
                -------------------
                 0 9 0|0 0 0|7 3 0
                 7 0 4|3 0 0|2 0 9
                 0 3 2|0 9 4|6 0 0


Errores: 1
Ingrese su respuesta:
```

Por el contrario al caso anterior, si el usuario ingresa valores no válidos (menores que 1 o mayores que 9), no se contará como error. Para la demostración de este proceso, se utilizó la herramienta GDB. Se creó un punto de ruptura, en la línea 41 para mostrar el valor que contiene la variable “errores” tras haber ingresado valores no válidos.

```
37
38          //Condicion para limitar que los valores enteros ingresados sean mayores que cero, y menores que 10.
39          if (x < 1 || x > 9 || y < 1 || y > 9 || z < 1 || z > 9) {
40            printf("Los valores ingresados no v\u00E1lidos\n");
(gdb) l
41            system("read -n 1 -s -p\"Presione cualquier tecla para continuar\""); //Espera a que el usuario presione alguna tecla para
 continuar con la ejecucion.
42          }
43
```

```
(gdb) b 41
Breakpoint 1 at 0x1403: file main.c, line 41.
(gdb)
```

```
(gdb) r
Starting program: /home/marko/Documents/EDAI/practica1/sudoku/sudoku

                *** Sudoku ***

Ingrese tres números enteros separados por una coma
-Deben ser numeros entre 1 y 9, incluyendo el 1 y 9
-El primer número corresponde al renglón
-El segundo número corresponde a la columna
-El tercer número corresponde a su respuesta


                 4 2 7|1 0 0|0 6 8
                 0 0 5|0 0 6|3 0 0
                 6 0 3|0 0 0|1 0 0
                -------------------
                 2 0 0|0 1 0|4 0 0
                 3 4 0|0 6 7|0 5 1
                 8 0 1|0 5 0|0 2 0
                -------------------
                 0 9 0|0 0 0|7 3 0
                 7 0 4|3 0 0|2 0 9
                 0 3 2|0 9 4|6 0 0


Errores: 0
Ingrese su respuesta: 0,0,0
Los valores ingresados no válidos

Breakpoint 1, main () at main.c:41
41            system("read -n 1 -s -p\"Presione cualquier tecla para continuar\""); //Espera a que el usuario presione alguna tecla para
 continuar con la ejecucion.
(gdb) p errores
$1 = 0
(gdb)
```

El juego terminará si el usuario completa el sudoku, para que el programa sepa que el usuario completó el sudoku , se suman todos los elementos del arreglo que contiene las respuestas del usuario, si el resultado es 405, el programa termina y muestra un mensaje de felicitación, por el contrario, si el resultado es diferente de 405, el código continua su ejecución. Al igual que en el caso anterior, se utilizó la herramienta GDB, con la que se creó un punto de ruptura en la línea número 73.

El resultado de la suma es un número diferente de 405.

```
73
74          system("clear"); //Limpia la pantalla ejecutando un script propio de la terminal de comandos.
75        }while( a != 405); //El ciclo se repite hasta que la suma de los elementos del arreglo respuestas sea 405.
76
```

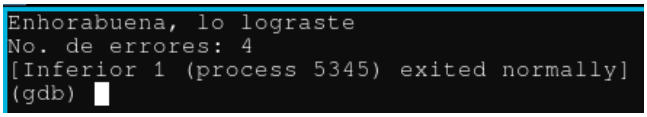
```
(gdb) b 73
Breakpoint 1 at 0x1524: file main.c, line 74.
(gdb)
```

```
(gdb) r
Starting program: /home/marko/Documents/EDAI/practica1/sudoku/sudoku

                *** Sudoku ***

Ingrese tres números enteros separados por una coma
-Deben ser numeros entre 1 y 9, incluyendo el 1 y 9
-El primer número corresponde al renglón
-El segundo número corresponde a la columna
-El tercer número corresponde a su respuesta


               4 2 7|1 0 0|0 6 8
               0 0 5|0 0 6|3 0 0
               6 0 3|0 0 0|1 0 0
              -------------------
               2 0 0|0 1 0|4 0 0
               3 4 0|0 6 7|0 5 1
               8 0 1|0 5 0|0 2 0
              -------------------
               0 9 0|0 0 0|7 3 0
               7 0 4|3 0 0|2 0 9
               0 3 2|0 9 4|6 0 0


Errores: 0
Ingrese su respuesta: 1,5,3

Breakpoint 1, main () at main.c:74
74          system("clear"); //Limpia la pantalla ejecutando un script propio de la terminal de comandos.
(gdb) p a
$1 = 172
(gdb)
```

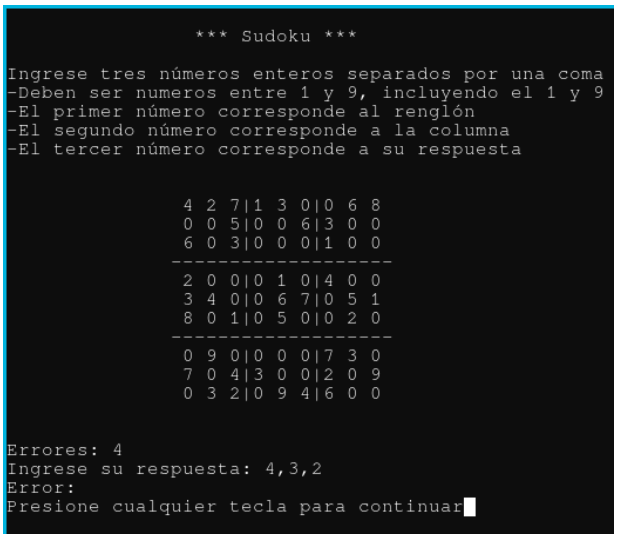
El resultado de la suma es igual a 405.

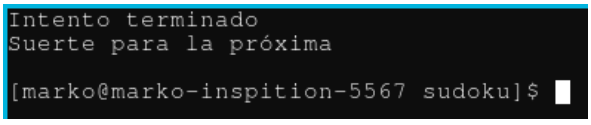
```
                *** Sudoku ***

Ingrese tres números enteros separados por una coma
-Deben ser numeros entre 1 y 9, incluyendo el 1 y 9
-El primer número corresponde al renglón
-El segundo número corresponde a la columna
-El tercer número corresponde a su respuesta


               4 2 7|1 3 9|5 6 8
               9 1 5|8 7 6|3 4 2
               6 8 3|5 4 2|1 9 7
              -------------------
               2 5 6|9 1 8|4 7 3
               3 4 9|2 6 7|8 5 1
               8 7 1|4 5 3|9 2 6
              -------------------
               5 9 8|6 2 1|7 3 4
               7 6 4|3 8 5|2 1 9
               1 3 2|7 9 4|6 8 0


Errores: 4
Ingrese su respuesta: ^C
Program received signal SIGINT, Interrupt.
0x00007ffff7ed6052 in read () from /usr/lib/libc.so.6
(gdb) b 74
Breakpoint 2 at 0x555555555524: file main.c, line 74.
(gdb) c
Continuing.
9,9,5

Breakpoint 2, main () at main.c:74
74          system("clear"); //Limpia la pantalla ejecutando un script propio de la terminal de comandos.
(gdb) p a
$2 = 405
(gdb)
```

Una vez se concluyó el sudoku, se muestra adicionalmente el número de errores cometidos durante su resolución.



Si el usuario comete 5 errores, el intento de resolver el sudoku termina automáticamente y se muestra un mensaje que notifica al usuario.





Conclusión

El uso de arreglos permite dar solución a problemas con un mayor grado de complejidad, permitiendo crear datos del mismo tipo que guarden relación entre si, en el caso del ejercicio 1, el uso de arreglos para realizar un sudoku permitió actualizar eficientemente en pantalla los valores correctos que eran ingresados por el usuario, así como, determinar hasta que momento el programa debía terminar.

Extra a lo solicitado en la práctica, el uso de librerías como "*stdlib.h*", "*string.h*", y la creación de librerías propias, permitía optimizar y añadir más funcionalidades al programa.

Referencias

Cohi (2009). Apple. "*How do I pause C command-line program?*". Recuperado el 14 de marzo de 2021, de https://discussions.apple.com/thread/2130719

Solano J. (2019). Laboratorio de Computación Salas A y B. "*Manual de prácticas del laboratorio de Estructuras de datos y algoritmos I*". Recuperado de http://lcp02.fi-b.unam.mx/

Sven (2010). Stackoverflow. "*How do I clear an array in C?*". Recuperado el 13 de marzo de 2021, de https://stackoverflow.com/questions/3831191/how-do-i-clear-an-array-in-c