

## EXPLICACION DE FUNCIONES

### Operación `mostrarLocal`:

Entrada: matriz y el número de pisos y locales.

Salida: ninguna.

Conceptos usados: paso de parámetro por referencia, ciclos con for para recorrer matriz, condicionales.

**Nuevo:** se le agrego para mostrar una nueva parte del struct.

### Operación `idLocal`:

Entrada: ninguna.

Salida: id aleatorio para completar la creación de un local.

Conceptos usados: uso de las librerías `#include <time.h>` y `#include <string.h>` para generar un numero aleatorio.

### Operación `menu`:

Entrada: ninguna.

Salida: el opc para que sea identificado como la selección que el usuario desee.

Conceptos usados: ningún concepto extraordinario.

**Nuevo:** se le agrego las opciones para entrar a las funciones de `ordenamientos`, `save` y `demo`.

### Operación `menuEdit`:

Entrada: ninguna.

Salida: opc como ayuda para la función `editarLocal`.

Conceptos usados: ningún concepto extraordinario.

**Nuevo:** se le agrego la opción para editar el `genero`.

### Operación `alquilarLocal`:

Entrada: matriz y el número de pisos y locales.

Salida: ninguna.

Conceptos usados: paso de parámetro por referencia, ciclo con `while` para impedir que una mala digitación dañe el programa, condicionales y el uso de `struct`.

**Nuevo:** Se agrego la opción para identificar el `genero` Además de usar la opción de `planosCC` para identificar si el CVA esta lleno o no.

### Operación `eliminarLocal`:

Entrada: matriz y el número de pisos y locales.

Salida: ninguna.

Conceptos usados: paso de parámetro por referencia ciclos con `while` para impedir que una mala digitación dañe el programa y el uso de condicionales.

### Operación `editarLocal`:

Entrada: matriz y el número de pisos y locales.

Salida: ninguna.

Conceptos usados: paso de parámetro por referencia ciclos con `for` doble para recorrer la matriz y condicionales, además de utilizar una función auxiliar la cual es `menuEdit`.

Nuevo: se le agrego la opción para editar el `genero`.

### Operación `TopVentas`:

Entrada: matriz y el número de pisos y locales.

Salida: ninguna.

Conceptos usados: paso de parámetro por referencia ciclos con `for` para recocer la matriz y condicionales.

### Operación `comprarEn`:

Entrada: matriz y el número de pisos y locales.

Salida: ninguna.

Conceptos usados: paso de parámetro por referencia ciclos con `for` para recocer la matriz y condicionales, el uso de `strcmp` de la Libreria `#include <string.h>` para realizar una comparación entre 2 valores.

### Operación gananciasCC:

Entrada: matriz y el número de pisos y locales.

Salida: ninguna.

Conceptos usados: paso de parámetro por referencia el uso de ciclo `for` más el uso de un valor de un `struct`.

### Operación planosCC:

Entrada: matriz y el número de pisos y locales.

Salida: ninguna.

Conceptos usados: paso de parámetro por referencia y ciclo `for` para recorrer la matriz y condicionales.

**Nuevo:** se cambió de `void` a `int` para ser usada de forma recursiva en la función `alquilarLocal`.

### Operación Reporte de Locales:

Entrada: matriz, el número de pisos y locales.

Salida: ninguna.

Conceptos usados: paso de parámetro por referencia y ciclo `for` para recorrer la matriz y condicionales más el manejo de archivos.

**Nuevo:** se le agrego para que pueda guardar el `genero` del local.

## NUEVAS DEL PROYECTO FINAL

### Operación `pruebasInt`:

Entrada: Variable con el número que se desea evaluar.

Salida: Ninguna.

Conceptos usados: El manejo de excepciones y recursión.

### Operación `generos`:

Entrada: Ninguna.

Salida: la opción almacenada en `opc`.

Conceptos usados: Ningún concepto extraordinario.

### Operación `ordenamientos`:

Entrada: Ninguna.

Salida: La opción seleccionada por el usuario almacenado por la variable `opc`.

Conceptos usados: Ningún concepto extraordinario.

Operación `intercambiar`:

Entrada: Punteros del struct.

Salida: Ninguna.

Conceptos usados: ordenamientos, Recursión para el uso de la función `particion` que a su vez esta de la función `quickSortAux` que a su vez esta de la función `quick_sort` y manejo de punteros.

Operación `particion`:

Entrada: array, entero de inicio del array y entero de fin del array.

Salida: entero

Conceptos usados: ordenamientos, recursión de la función `quickSortAux` que a su vez esta de la función `quick_sort` el uso de ciclo `while` e `if else`.

Operación `merge`:

Entrada: puntero de array un entero de inicio, la longitud del arreglo y la división del array en partes.

Salida: Ninguna.

Conceptos usados: ordenamientos, Recursión para el uso de la función `mergeSortAux` que a su vez esta de la función `merge_sort`, manejo de ciclo `for` y `while` y manejo de punteros.

### Operación `mergeSortAux`:

Entrada: puntero de array, un entero de inicio y la longitud del arreglo.

Salida: Ninguna.

Conceptos usados: ordenamientos, Recursión consigo misma y para el uso de la función `merge_sort`.

### Operación `quickSortAux`:

Entrada: puntero de array, un entero de inicio y entero con el final del arreglo.

Salida: Ninguna.

Conceptos usados: ordenamientos, recursión consigo misma, `partition` y para el uso de la función `quick_sort`.

Operación `selection_sort`:

Entrada: matriz y el número de pisos y locales.

Salida: Ninguna.

Conceptos usados: ordenamientos, ciclos `for` y el uso del `if`.

Operación `insert_sort`:

Entrada: matriz y el número de pisos y locales.

Salida: Ninguna.

Conceptos usados: ordenamientos, ciclos `for`, ciclo `while` y el uso del `if`.

Operación `merge_sort`:

Entrada: matriz y el número de pisos y locales.

Salida: Ninguna.

Conceptos usados: ordenamientos, ciclos `for` y el uso de recursión con la función `mergeSortAux`.

Operación `quick_sort`:

Entrada: matriz y el número de pisos y locales.

Salida: Ninguna.

Conceptos usados: ordenamientos, ciclos `for` y el uso de recursión con la función `quickeSortAux`.