

**UNIVERSIDAD DE SAN CARLOS DE GUATEMALA.**  
**CENTRO UNIVERSITARIO DE OCCIDENTE.**  
**DIVISIÓN DE CIENCIAS DE LA INGENIERÍA.**  
**INGENIERÍA EN CIENCIAS Y SISTEMAS.**  
**LABORATORIO DE ESTRUCTURA DE DATOS.**

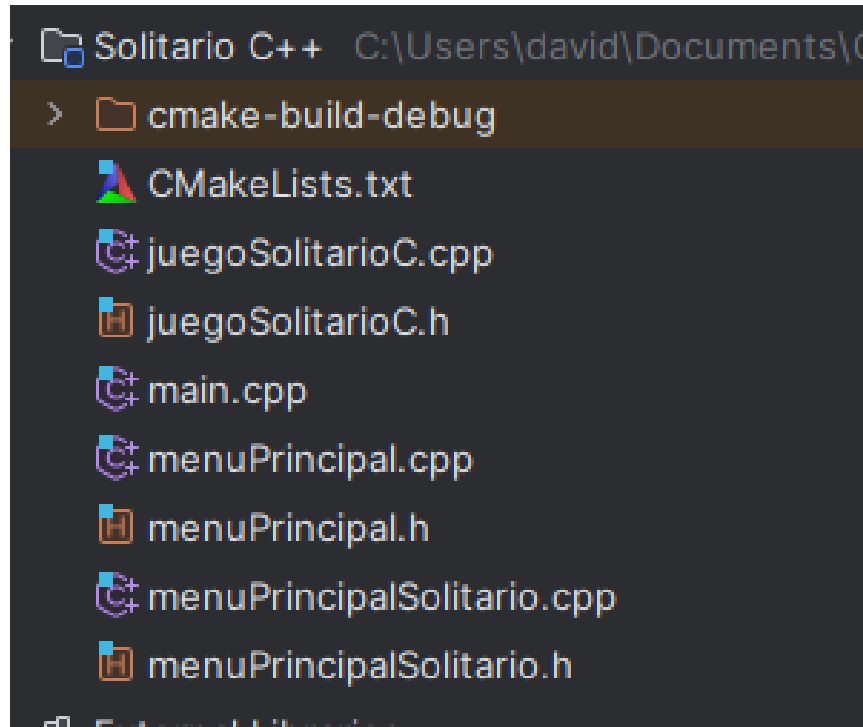


**PROYECTO 01.**  
**SOLITARIO C++.**  
**MANUAL TECNICO.**

**ESTUARDO DAVID BARRENO NIMATUJ.**  
**CARNÉ: 201830233.**

## METODOS UTILIZADOS.

## CREACION DE LAS CLASES.



Para la creación del proyecto se utilizaron las siguientes clases.

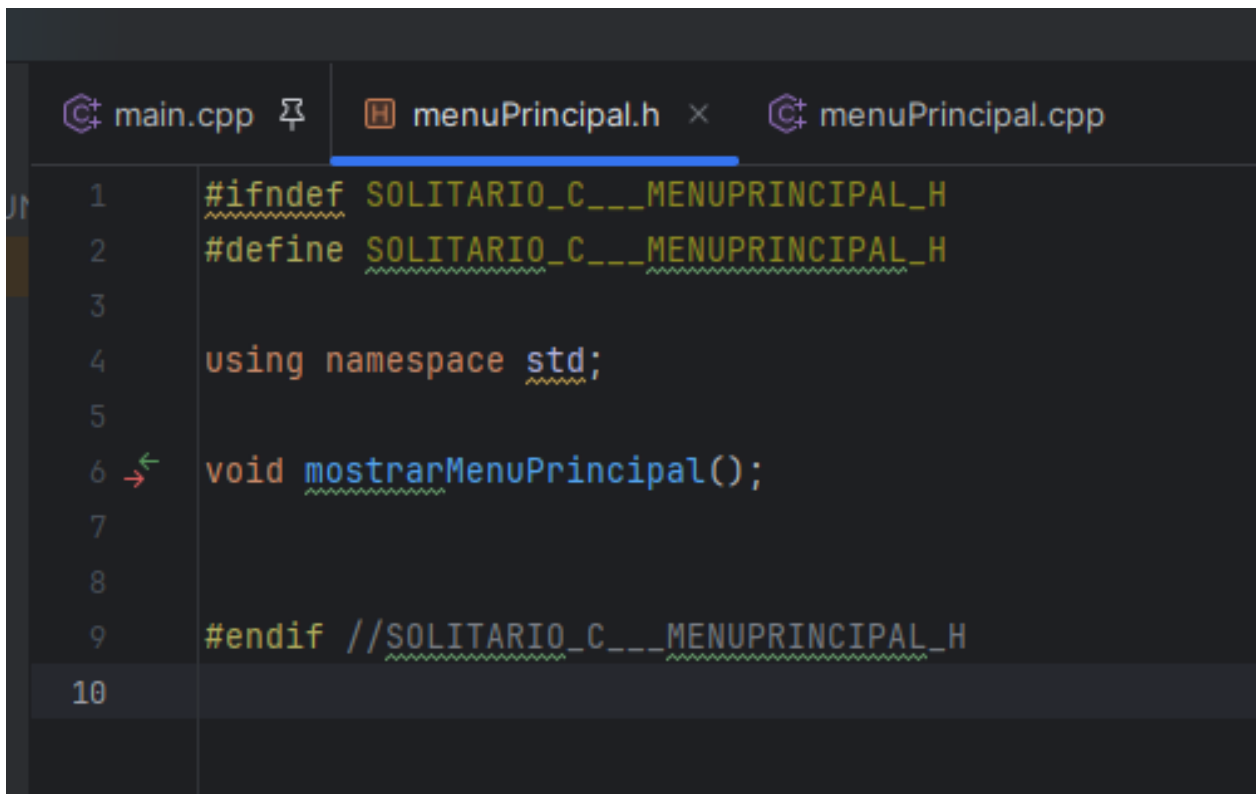
La clase main: es la encargada de invocar el método principal del programa.

```
main.cpp
1  #include "menuPrincipal.h"
2
3  int main() {
4
5      mostrarMenuPrincipal();
6
7      return 0;
8  }
9
```

Cada vez que se crean mas clases, se generan 2 archivos “menuPrincipal.h” y “menuPrincipal.cpp”.

En las clases.h usualmente van los atributos privados y las clases tipo publicas de los métodos de su clase cpp, que se usaran para pasar atributos o llamar funciones entre clases.

En mi caso solo esto llamando a la funcoin “mostrarMenuPrincipal” que esta ubicado en mi clase cpp.



```
1  #ifndef SOLITARIO_C__MENUPRINCIPAL_H
2  #define SOLITARIO_C__MENUPRINCIPAL_H
3
4  using namespace std;
5
6  void mostrarMenuPrincipal();
7
8
9  #endif //SOLITARIO_C__MENUPRINCIPAL_H
10
```

El método que llamo es directamente un menú, al no pasar parámetros o valores, simplemente se puede llamar a la función.

```
main.cpp  menuPrincipal.h  menuPrincipal.cpp x
1  #include "menuPrincipal.h"
2  #include "menuPrincipalSolitario.h"
3  #include <iostream>
4  #include <limits> // Para limpiar el buffer de entrada
5  using namespace std;
6
7  void mostrarMenuPrincipal() {
8      int opcion;
9
10     do {
11         cout << "\n\n ----- \n";
12         cout << "   --- BIENBENIDO AL JUEGO   --- \n";
13         cout << "   --- SOLITARIO EN C++   --- \n";
14         cout << "   ----- \n";
15         cout << "\n\nA continuacion seleccione una opcion: \n\n";
16         cout << "1--- Jugar Solitario en C++. \n";
17         cout << "2--- Soporte Tecnico. \n";
18         cout << "3--- Salir del sistema. \n\n";
19         cout << "Su respuesta es: --->";
20
21         try {
22             std::cin >> opcion;
```

## CREACION DE LOS MENUS.

Los menús utilizan una estructura Try - Catch, en donde se le define las opciones necesarias y el manejo de errores que se puedan presentar, cada menú se adaptó según las necesidades del programa.

```

do {
    cout << "\n\n ----- \n";
    cout << "    ---  BIENBENIDO AL JUEGO    --- \n";
    cout << "    ---  SOLITARIO EN C++    --- \n";
    cout << "    ----- \n";
    cout << "\n\nA continuacion seleccione una opcion: \n\n";
    cout << "1--- Jugar Solitario en C++. \n";
    cout << "2--- Soporte Tecnico. \n";
    cout << "3--- Salir del sistema. \n\n";
    cout << "Su respuesta es: --->";

    try {
        std::cin >> opcion;

        if (std::cin.fail()) {
            throw std::invalid_argument("ERROR --- Ingresar solo numeros del 1 al 3.");
        }

        switch (opcion) {
            case 1:
                mostrarMenuPrincipalSolitario();
                break;
            case 2:
                std::cout << "Hecho por 201830233... \n";
                break;
            case 3:
                std::cout << "\n\n ----- \n";
                std::cout << "    ---  SALIENDO DEL SISTEMA    --- \n";
                std::cout << "    ---  De parte de nuestros Administradores    --- \n";
                std::cout << "    ---  le deseamos un lindo dia!!    --- \n";
                std::cout << "    ----- \n";
                break;
            default:
                std::cout << "ERROR --- Ingresar solo numeros del 1 al 3. \n";
        }
    } catch (const std::invalid_argument& e) {
        std::cout << e.what() << "\n";
        std::cin.clear();
        std::cin.ignore(n: std::numeric_limits<std::streamsize>::max(), delim: '\n');
    }

    } while (opcion != 3);
}

```

## CREACION DEL MAZO DE CARTAS.

Para la creación del mazo de cartas utilizamos una cadena de 52 strings , donde cada subcadena de strings contiene 1 de las 52 cartas disponibles en el mazo.

```
void mostrarJuegoSolitarioC() {  
    string cartasOrigen[] = { [0]: "01-<3-R", [1]: "02-<3-R", [2]: "03-<3-R", [3]: "04-<3-R", [4]: "05  
                             [13]: "01-<>-R", [14]: "02-<>-R", [15]: "03-<>-R", [16]: "04-<>-R", [17]: "05  
                             [26]: "01-E3-N", [27]: "02-E3-N", [28]: "03-E3-N", [29]: "04-E3-N", [30]: "05  
                             [39]: "01-!!-N", [40]: "02-!!-N", [41]: "03-!!-N", [42]: "04-!!-N", [43]: "05
```

Las cartas se clasificaron en este orden “Numero” – “Tipo” – “Color”

Se uso el “-“ entre atributos de las cartas para facilitar su manipulación para definir las condiciones que se usaran después.

```
#include <algorithm> //Solo se uso para ordenar las cartas al azar  
#include <random>    //Solo se uso para ordenar las cartas al azar
```

Aclaro que únicamente se utilizaron las librerías aalgorithm y random, para la creación del mazo al azar.

```
const int tamaño = sizeof(cartasOrigen) / sizeof(cartasOrigen[0]);  
  
string cartasOrigen2[tamaño];  
copy( first: begin( &: cartasOrigen), last: end( &: cartasOrigen), result: begin( &: cartasOrigen2));  
  
random_device rd;  
mt19937 g( sd: rd());  
shuffle( first: begin( &: cartasOrigen2), last: end( &: cartasOrigen2), &: g);  
  
for (const auto& elemento :const string& : cartasOrigen2) {  
    //cout << elemento << endl;    //----- Ya imprime las cartas al azar :D  
}  
  
//cout<<"ACA esta"+cartasOrigen2[51];
```

Utilizamos un pequeño método para usar la cadena de strings de las cartas “cartasOrigen” e insertarlas en un orden aleatorio en “cartasOrigen2”.

## CREACION DE LAS COLAS / MAZOS DE CARTAS.

Para la creación de la agrupación de los mazos utilizamos “Colas” principalmente 2 Colas.

- 1 Cola para el mazo de cartas en uso.
- 1 Cola para el mazo de cartas en reposo.

Para llenar el mazo de cartas en uso utilizamos el pequeño arreglo

```
int contBarajaInicial;  
for (contBarajaInicial = 1; contBarajaInicial <= 52; contBarajaInicial++){  
    insertarColaBarajaInicial( & frenteBarajaInicial, & finBarajaInicial, nBarajaInicial: cartasOrigen2[contBarajaInicial]);  
}
```

Que nos va agregando las 52 cartas ordenadas previamente al azar en nuestra cola.

**Para agregar o insertar los elementos en nuestra cola seguimos 3 simples pasos:**

- 1-Crear el espacio en memoria para almacenar un nodo.
- 2-Asignar ese nuevo nodo al dato que queremos insertar.
- 3-Asignar los punteros frente y fin hacia el nuevo nodo.

**Para eliminar los elementos en nuestra cola seguimos 3 simples pasos:**

- 1-Obtener el valor del nodo.
- 2-Crear un nodo aux y asignarle el frente de la cola.
- 3-Eliminar el nodo del frente de la cola.

Estructura:

```
struct NodoBarajaInicial{  
    string barajaInicial;  
    NodoBarajaInicial *siguienteBarajaInicial;  
};
```

Prototipo de las funciones:

```
void insertarColaBarajaInicial(NodoBarajaInicial *&, NodoBarajaInicial *&, string);  
bool colaBarajaInicial_vacia(NodoBarajaInicial *);  
void eliminarColaBarajaInicial(NodoBarajaInicial *&, NodoBarajaInicial *&, string&);
```

Función para insertar elementos en la cola1:

```
void insertarColaBarajaInicial(NodoBarajaInicial *&frenteBarajaInicial, NodoBarajaInicial *&finBarajaInicial, string nBarajaInicial){  
    NodoBarajaInicial *nuevoNodoBarajaInicial = new NodoBarajaInicial();  
  
    nuevoNodoBarajaInicial->barajaInicial = nBarajaInicial;  
    nuevoNodoBarajaInicial->siguienteBarajaInicial = NULL;  
    if (colaBarajaInicial_vacia(frenteBarajaInicial)){  
        frenteBarajaInicial = nuevoNodoBarajaInicial;  
    }  
    else{  
        finBarajaInicial->siguienteBarajaInicial = nuevoNodoBarajaInicial;  
    }  
    finBarajaInicial = nuevoNodoBarajaInicial;  
    //cout<<"Elemento " <<nBarajaInicial<<" Insertado correctamente\n";  
}
```

Función para eliminar elementos en la cola1:

```
void eliminarColaBarajaInicial(NodoBarajaInicial *&frenteBarajaInicial, NodoBarajaInicial *&finBarajaInicial, string &nBarajaInicial){  
    nBarajaInicial = frenteBarajaInicial->barajaInicial;  
    NodoBarajaInicial *aux = frenteBarajaInicial;  
  
    if(frenteBarajaInicial == finBarajaInicial){  
        frenteBarajaInicial = NULL;  
        finBarajaInicial = NULL;  
    } else{  
        frenteBarajaInicial = frenteBarajaInicial -> siguienteBarajaInicial;  
    }  
    delete aux;  
}
```



Funcion auxiliar para saber si la cola1 esta vacia:

```
bool colaBarajaInicial_vacia(NodoBarajaInicial *frenteBarajaInicial){  
    return (frenteBarajaInicial == NULL)? true : false;  
}
```

Prácticamente repetimos el proceso para la cola2 cuando lo necesitemos.

### **CREACION DE LAS PILAS / COLUMNAS DE LAS CARTAS.**

Para la creación de las pilas creamos un total de 14 pilas,

2 pilas por cada columna, la razón es simple ya que tenemos un mazo con cartas “ocultas” y con cartas “visibles” en cada columna de la A - G, opte por usar una pila de cartas ocultas para la posición en columna A y una pila de cartas visibles para la misma posición, el proceso se repitió hasta completar las 14 pilas.

**Para insertar elementos en una pila seguimos los siguientes pasos:**

- 1-Crear el espacio en memoria para almacenar un nodo.
- 2-Cargar el valor dentro del nodo(dato).
- 3-Cargar el puntero pila dentro del nodo (\*siguiente).
- 4-Asignar el nuevo nodo a pila.

**Para eliminar elementos en una pila seguimos los siguientes pasos:**

- 1-Crear una variable \*auxiliar de tipo Nodo.
- 2-Igualar el n a auxiliar->dato.
- 3-Pasar pila a siguiente nodo.
- 4-Eliminar Auxiliar.

Estructuras: de las 14 pilas, 1 visible, 1 oculta.

```
▼ struct NodoBarajaOcultaB{  
    string barajaOcultaB;  
    NodoBarajaOcultaB *siguienteBarajaOcultaB;  
};  
  
▼ struct NodoBarajaVisibleB{  
    string barajaVisibleB;  
    NodoBarajaVisibleB *siguienteBarajaVisibleB;  
};
```

Prototipo de funciones: de las 14 pilas, 1 visible, 1 oculta.

```
void insertarPilaBarajaOcultaB(NodoBarajaOcultaB *&, string);  
void insertarPilaBarajaOcultaC(NodoBarajaOcultaC *&, string);  
void insertarPilaBarajaOcultaD(NodoBarajaOcultaD *&, string);  
void insertarPilaBarajaOcultaE(NodoBarajaOcultaE *&, string);  
void insertarPilaBarajaOcultaF(NodoBarajaOcultaF *&, string);  
void insertarPilaBarajaOcultaG(NodoBarajaOcultaG *&, string);  
  
void insertarPilaBarajaVisibleA(NodoBarajaVisibleA *&, string);  
void insertarPilaBarajaVisibleB(NodoBarajaVisibleB *&, string);  
void insertarPilaBarajaVisibleC(NodoBarajaVisibleC *&, string);  
void insertarPilaBarajaVisibleD(NodoBarajaVisibleD *&, string);  
void insertarPilaBarajaVisibleE(NodoBarajaVisibleE *&, string);  
void insertarPilaBarajaVisibleF(NodoBarajaVisibleF *&, string);  
void insertarPilaBarajaVisibleG(NodoBarajaVisibleG *&, string);
```

Funcion agregar elementos a una pila: de las 14 pilas, 1 visible,  
1 oculta.

```
void insertarPilaBarajaOcultas(NodoBarajaOcultas *&pilaNodoBarajaOcultas, string nBarajaOcultas){
    NodoBarajaOcultas *nuevoNodoBarajaOcultas = new NodoBarajaOcultas();
    nuevoNodoBarajaOcultas->barajaOcultas = nBarajaOcultas;
    nuevoNodoBarajaOcultas->siguienteBarajaOcultas = pilaNodoBarajaOcultas;
    pilaNodoBarajaOcultas = nuevoNodoBarajaOcultas;
}
```

```
void insertarPilaBarajaVisible(NodoBarajaVisible *&pilaNodoBarajaVisible, string nBarajaVisible){
    NodoBarajaVisible *nuevoNodoBarajaVisible = new NodoBarajaVisible();
    nuevoNodoBarajaVisible->barajaVisible = nBarajaVisible;
    nuevoNodoBarajaVisible->siguienteBarajaVisible = pilaNodoBarajaVisible;
    pilaNodoBarajaVisible = nuevoNodoBarajaVisible;
}
```

### CREACION DE LA MATRIZ.

```
-----
  A   |   B   |   C   |   D   |   E   |   F   |   G   |
-----
10-!!-N |Sin cartas
-----
01-<3-R |  *      |  *      |  *      |  *      |  *      |  *      |
      02-<3-R |  *      |  *      |  *      |  *      |  *      |  *      |
              J-<3-R |  *      |  *      |  *      |  *      |  *      |
              04-!!-N |  *      |  *      |  *      |  *      |  *      |
                  09-E3-N |  *      |  *      |  *      |  *      |  *      |
                      Q-!!-N |  *      |  *      |  *      |  *      |  *      |
                          10-<>-R |  *      |  *      |  *      |  *      |  *      |
-----
```

- 1--- Siguiente carta.
- 2-- Insertar carta.
- 3--- Mover carta.
- 4--- Regresar movimiento.
- 5--- Pista.
- 6--- RENDIRSE.

```
-----
Su respuesta es: --->
```

Se utilizo una matriz como forma de representar las cartas y la interfaz de juego.

```
int ocultoA = 0;
int ocultoB = 0;
int ocultoC = 0;
int ocultoD = 0;
int ocultoE = 0;
int ocultoF = 0;
int ocultoG = 0;

int visivleA = 0;
int visivleB = 0;
int visivleC = 0;
int visivleD = 0;
int visivleE = 0;
int visivleF = 0;
int visivleG = 0;

string matrizA[7][7];
```

Primero generamos los contadores para las pilas ocultas y las pilas visibles de cada columna. Esto nos ayudara después para manipular los datos.

La forma que se utlizo para la creación de la matriz fue con manipulación directa de pilas y colas:

- Se crean 14 variables contadoras, 7 para las cartas visibles y 7 para las cartas ocultas.

- Se crea una matriz base de 7\*7, esta se podrá ir cambiando de tamaño después con matrices auxiliares y sobreescritura de datos.

- La condición de barajear las cartas es:

- A --- Tiene 0 cartas ocultas y 1 carta visible.
- B --- Tiene 1 carta oculta y 1 carta visible.
- C --- Tiene 2 cartas ocultas y 1 carta visible.
- D --- Tiene 3 cartas ocultas y 1 carta visible.

E --- Tiene 4 cartas ocultas y 1 carta visible.

F --- Tiene 5 cartas ocultas y 1 carta visible.

G --- Tiene 6 cartas ocultas y 1 carta visible.

Esa es la razón de usar una matriz base de 7\*7.

Se puede visualizar así:

```
A --- B --- C --- D --- E --- F --- G
1 --- * --- * --- * --- * --- * --- *
   --- 1 --- * --- * --- * --- * --- *
   ---   --- 1 --- * --- * --- * --- *
   ---   ---   --- 1 --- * --- * --- *
   ---   ---   ---   --- 1 --- * --- *
   ---   ---   ---   ---   --- 1 --- *
   ---   ---   ---   ---   ---   --- 1
```

Mi método fue insertar de un solo tanto las cartas en el mazo , en la matriz y en las pilas y colas.

A --- Tiene 0 cartas ocultas y 1 carta visible.

```
insertarPilaBarajaVisibleA( &: pilaNodoBarajaVisibleA, nBarajaVisibleA: finBarajaInicial->barajaInicial);visivleA++;
matrizA[0][0] = finBarajaInicial->barajaInicial+" | ";
```

El resumen es prácticamente de la cola con las 52 cartas, la primera en Salir se pasa a la matriz, a su vez se escribe en la cola que corresponde. Y se elimina de una vez el dato en la cola, así no hay duplicados y repartimos correctamente las 52 cargas.

Se puede observar que tenemos 7 filas, cada una con un orden descendiente de datos así que podemos usarlo de la siguiente manera:

```

insertarPilaBarajaVisibleA( &. pilaNodoBarajaVisibleA, nBarajaVisibleA: finBarajaInicial->barajaInicial);visivleA++;
matrizA[0][0] = finBarajaInicial->barajaInicial+" | ";
eliminarColaBarajaInicial( &. frenteBarajaInicial, &. finBarajaInicial, &. finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultoB( &. pilaNodoBarajaOcultoB, nBarajaOcultoB: finBarajaInicial->barajaInicial);ocultoB++;
matrizA[0][1] = "***** | ";
eliminarColaBarajaInicial( &. frenteBarajaInicial, &. finBarajaInicial, &. finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultoC( &. pilaNodoBarajaOcultoC, nBarajaOcultoC: finBarajaInicial->barajaInicial);ocultoC++;
matrizA[0][2] = "***** | ";
eliminarColaBarajaInicial( &. frenteBarajaInicial, &. finBarajaInicial, &. finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultoD( &. pilaNodoBarajaOcultoD, nBarajaOcultoD: finBarajaInicial->barajaInicial);ocultoD++;
matrizA[0][3] = "***** | ";
eliminarColaBarajaInicial( &. frenteBarajaInicial, &. finBarajaInicial, &. finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultoE( &. pilaNodoBarajaOcultoE, nBarajaOcultoE: finBarajaInicial->barajaInicial);ocultoE++;
matrizA[0][4] = "***** | ";
eliminarColaBarajaInicial( &. frenteBarajaInicial, &. finBarajaInicial, &. finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultoF( &. pilaNodoBarajaOcultoF, nBarajaOcultoF: finBarajaInicial->barajaInicial);ocultoF++;
matrizA[0][5] = "***** | ";
eliminarColaBarajaInicial( &. frenteBarajaInicial, &. finBarajaInicial, &. finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultoG( &. pilaNodoBarajaOcultoG, nBarajaOcultoG: finBarajaInicial->barajaInicial);ocultoG++;
matrizA[0][6] = "***** | ";
eliminarColaBarajaInicial( &. frenteBarajaInicial, &. finBarajaInicial, &. finBarajaInicial->barajaInicial);

```

```

insertarPilaBarajaVisibleB( &: pilaNodoBarajaVisibleB, nBarajaVisibleB: finBarajaInicial->barajaInicial);visibleB++;
matrizA[1][0] = "          ";
matrizA[1][1] = finBarajaInicial->barajaInicial+" | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultaC( &: pilaNodoBarajaOcultaC, nBarajaOcultaC: finBarajaInicial->barajaInicial);ocultoC++;
matrizA[1][2] = "***** | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultaD( &: pilaNodoBarajaOcultaD, nBarajaOcultaD: finBarajaInicial->barajaInicial);ocultoD++;
matrizA[1][3] = "***** | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultaE( &: pilaNodoBarajaOcultaE, nBarajaOcultaE: finBarajaInicial->barajaInicial);ocultoE++;
matrizA[1][4] = "***** | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultaF( &: pilaNodoBarajaOcultaF, nBarajaOcultaF: finBarajaInicial->barajaInicial);ocultoF++;
matrizA[1][5] = "***** | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultaG( &: pilaNodoBarajaOcultaG, nBarajaOcultaG: finBarajaInicial->barajaInicial);ocultoG++;
matrizA[1][6] = "***** | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);

```

-La 3era fila:

```
insertarPilaBarajaVisibleC( &: pilaNodoBarajaVisibleC, nBarajaVisibleC: finBarajaInicial->barajaInicial);visivleC++
matrizA[2][0] ="          ";
matrizA[2][1] ="          ";
matrizA[2][2] = finBarajaInicial->barajaInicial+" | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultaD( &: pilaNodoBarajaOcultaD, nBarajaOcultaD: finBarajaInicial->barajaInicial);ocultoD++;
matrizA[2][3] ="***** | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultaE( &: pilaNodoBarajaOcultaE, nBarajaOcultaE: finBarajaInicial->barajaInicial);ocultoE++;
matrizA[2][4] ="***** | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultaF( &: pilaNodoBarajaOcultaF, nBarajaOcultaF: finBarajaInicial->barajaInicial);ocultoF++;
matrizA[2][5] ="***** | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultaG( &: pilaNodoBarajaOcultaG, nBarajaOcultaG: finBarajaInicial->barajaInicial);ocultoG++;
matrizA[2][6] ="***** | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
```

-La 4ta fila:

```
insertarPilaBarajaVisibleD( &: pilaNodoBarajaVisibleD, nBarajaVisibleD: finBarajaInicial->barajaInicial);visivleD++
matrizA[3][0] ="          ";
matrizA[3][1] ="          ";
matrizA[3][2] ="          ";
matrizA[3][3] = finBarajaInicial->barajaInicial+" | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultaE( &: pilaNodoBarajaOcultaE, nBarajaOcultaE: finBarajaInicial->barajaInicial);ocultoE++;
matrizA[3][4] ="***** | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultaF( &: pilaNodoBarajaOcultaF, nBarajaOcultaF: finBarajaInicial->barajaInicial);ocultoF++;
matrizA[3][5] ="***** | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultaG( &: pilaNodoBarajaOcultaG, nBarajaOcultaG: finBarajaInicial->barajaInicial);ocultoG++;
matrizA[3][6] ="***** | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
```

-La 5ta fila:

```
insertarPilaBarajaVisibleE( &: pilaNodoBarajaVisibleE, nBarajaVisibleE: finBarajaInicial->barajaInicial);visivleE++
matrizA[4][0] ="          ";
matrizA[4][1] ="          ";
matrizA[4][2] ="          ";
matrizA[4][3] ="          ";
matrizA[4][4] = finBarajaInicial->barajaInicial+" | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultaF( &: pilaNodoBarajaOcultaF, nBarajaOcultaF: finBarajaInicial->barajaInicial);ocultoF++;
matrizA[4][5] ="***** | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultaG( &: pilaNodoBarajaOcultaG, nBarajaOcultaG: finBarajaInicial->barajaInicial);ocultoG++;
matrizA[4][6] ="***** | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
```

-La 6ta fila:

```
insertarPilaBarajaVisibleF( &: pilaNodoBarajaVisibleF, nBarajaVisibleF: finBarajaInicial->barajaInicial);visivleF++;
matrizA[5][0] ="          ";
matrizA[5][1] ="          ";
matrizA[5][2] ="          ";
matrizA[5][3] ="          ";
matrizA[5][4] ="          ";
matrizA[5][5] = finBarajaInicial->barajaInicial+" | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
insertarPilaBarajaOcultaG( &: pilaNodoBarajaOcultaG, nBarajaOcultaG: finBarajaInicial->barajaInicial);ocultoG++;
matrizA[5][6] ="***** | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
```

-La 7ma fila:

```
insertarPilaBarajaVisibleG( &: pilaNodoBarajaVisibleG, nBarajaVisibleG: finBarajaInicial->barajaInicial);visivleG++;
matrizA[6][0] ="          ";
matrizA[6][1] ="          ";
matrizA[6][2] ="          ";
matrizA[6][3] ="          ";
matrizA[6][4] ="          ";
matrizA[6][5] ="          ";
matrizA[6][6] = finBarajaInicial->barajaInicial+" | ";
eliminarColaBarajaInicial( &: frenteBarajaInicial, &: finBarajaInicial, &: finBarajaInicial->barajaInicial);
```

Una vez terminado, tendremos construida nuestra matriz con las 52 cartas ya repartidas, nuestra cola con las cartas sobrantes y las pilas con sus respectivas cartas visibles y ocultas.