INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

REPORTE DE HANOI

Unidad de Aprendizaje: Fundamentos de Programación

Profesor: De Luna Caballero Roberto

Alumna: Montes Jaimes Danna Xiomara

Introducción

En el presente proyecto se documenta el desarrollo de la creación y resolución de un hanoi con interfaz gráfica diseñado en el lenguaje de programación c, para ello se comienza con una investigación previa sobre el lenguaje para conocerlo a profundidad, desde su sintaxis hasta sus aplicaciones y tipos de programación que puede llevar a cabo, en este caso parte desde la estructurada, pasando por orientada a objetos hasta algo avanzado.

También se indagó sobre la implementación de los gráficos (originarios del lenguaje c++) en el proyecto para el diseño de la interfaz en una ventana gráfica, así mismo se planeó el diseño de la plantilla base del hanoi que cuenta con tres torres y de uno a 10 discos como límite para elección del usuario al momento de compilar, todo desde cero mediante el uso de coordenadas tomando en cuenta las medidas de la pantalla de la computadora y las ventanas que se abrirán (consola y gráfica).

Pasando a la investigación teórica del proyecto se conoce que hoy en día la tecnología se encuentra sumamente globalizada, detrás de todos los programas y sistemas informáticos que se conocen y usan de manera cotidiana para facilitar diversas actividades de nuestro entorno, existe todo un proceso para poderlos crear, es el proceso de la programación, por medio de ella se establecen los pasos a seguir para la creación del código fuente de todos los programas. La programación se guía por una serie de normas y conjuntos de órdenes, instrucciones y expresiones que tienden a ser semejantes a una lengua natural acotada. Por lo cual se le asigna el nombre de lenguaje y existen varios tipos de ellos que son usados de acuerdo con lo que se planee realizar, cada uno cumple con ciertas especificaciones que lo hace diferente o similar a otros lenguajes y que en ocasiones les permite usarse de forma conjunta.

La programación ha sido la causante de que la tecnología haya podido avanzar hasta como se encuentra en la actualidad, permitiendo de cierta forma el desarrollo de inventos que facilitan las tareas que realizamos día con día. Tan solo durante los primeros años, las ideas aparecían de forma lenta y costaba desarrollarlas, planificarlas y plasmarlas, sin embargo, actualmente se cuenta con demasiada información en diferentes medios, como foros, medios de navegación, revistas, libros, entre otros medios digitales.

Se considera que la programación es la base del futuro, la encargada de que la tecnología se siga desarrollando y de que aparezcan inventos nuevos, algo tan simple como el juego de hanoi tiene una complejidad a un nivel increíble para un desarrollador, que para poder resolverlo toma en cuenta la implicación y el análisis matemático (para calcular el número de movimientos a realizar) hasta una comprensión y conocimiento de los conceptos de programación para realizar un código que sea eficiente y pueda completar de forma adecuada el planteamiento

inicial, todo esto se encuentra desarrollado de forma detallada en la continuación del reporte.

Marco Teórico

Lenguaje C

- **Historia:** Creado a mediados de los 70s por Brian Kerninghan y Dennis Ritchie como resultado de desarrollo de un lenguaje anterior, el BCPL, mediante el cual se desarrolló el lenguaje B, un antecedente directo del lenguaje C.

- **Características:**

  → Lenguaje estructurado exceptuando los bloques (no es posible declarar subrutinas)
  → No es de norma rígida en la comprobación de tipos de datos, ya que permite la conversión y la asignación entre varios tipos de datos diferentes.
  → Posee flexibilidad para la programación.
  → Tiene una baja comprobación de incorrecciones, siendo así que es responsabilidad del programador tener una buena sintaxis, lógica y código limpio para evitar errores, acciones que otros lenguajes realizan por si mismos.
  → Errores en los arrays ya que no comprueba el índice de referencia de estos, siendo así que muchas veces este sobrepasa el tamaño establecido.

- **Características básicas de un programa en C:**

  → Conjunto de funciones.
  → Librerías.
  → Prototipo de funciones.
  → Return 0.
  → Función llamada main (es la primera al ejecutarse al iniciar el programa), desde esta se mandan a llamar al resto de las funciones.

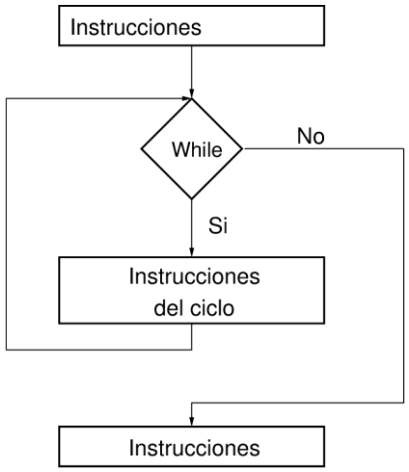
- **Standard ANSI-C:** Es un conjunto de 32 palabras reservadas definidas.

auto	Break	Case	Char	Const	Continue	Default
Do	Double	Else	Enum	Extern	Float	For

goto	If	Int	Long	Register	Retun	Short
signed	Sizeof	static	Struct	switch	typedef	union
unsigned	void	volatile	while			

- **Ventajas de programar en C:** El lenguaje C usa un lenguaje de máquina, por lo que es muy eficiente, con instrucciones muy optimizadas. Además, se puede llegar a pasar el código a otros sistemas operativos. Programar en C, da una eficiencia y capacidad de manejar todos los aspectos de las instrucciones del CPU. Posee mucha eficiencia, seguridad y optimización. Además de no ser un lenguaje sencillo su dificultad se ve reducida gracias a un código perfecto, una estructura limpia, lo que hace que el programador consiga crear aplicaciones de una manera rápida y potente.

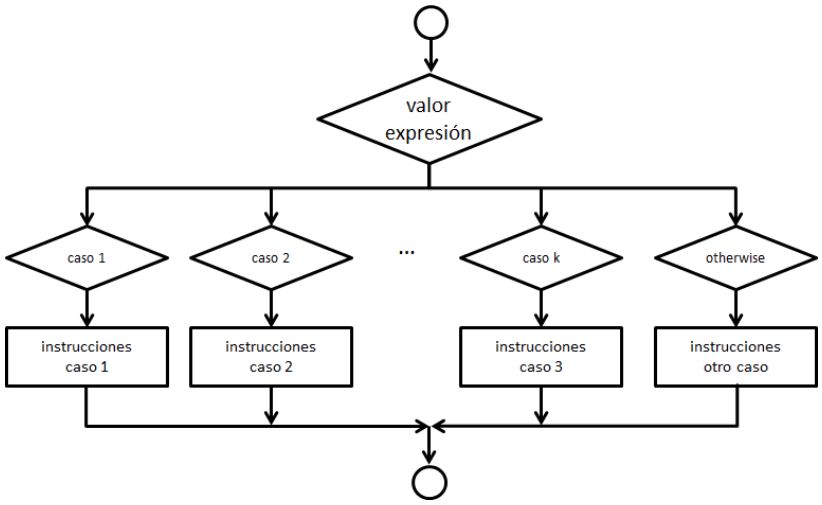
- **Funciones utilizadas en el proyecto:**
  → Ciclo while: Ejecuta las sentencias del bucle mientras una expresión es verdadera. Tiene la condición de salida al comienzo de la estructura por lo que las sentencias se ejecutarían de 0 a n veces. Ejemplo while{ }




```
while(condición){

  <listas de instrucciones condición=true>;
  <actualización de la condición>;

}
```

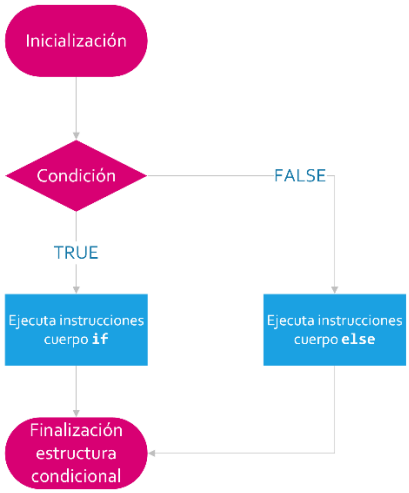
  → **Sentencia switch-case:** Proporciona la estructura selectiva múltiple. Evalúa una expresión de tipo entero y comprueba su

valor con cada una de las cláusulas case; si coincide ejecuta la sentencia y sigue la comprobación, en caso contrario pasa a la siguiente cláusula, en la parte de default incluirá las sentencias que se ejecutan si fallan todas las comprobaciones anteriores.



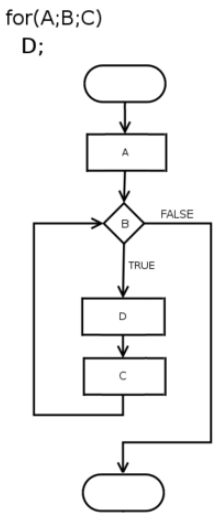

```
switch (val) {
        case val1:
                // instructions si val=val1
                break;
        case val2:
                // instructions si val=val2
                break;
        default:
                // instructions par défaut
                break;
}
```

→ **Sentencia if-else:** Se trata de una estructura selectiva simple y doble que maneja una condición principal a ser considerada, de esta dependerá el transcurso de la operación en el código.




```
if(condition){
        <liste d'instructions si condition=true>;
}else{
        <liste d'instructions si condition=false>;
}
```

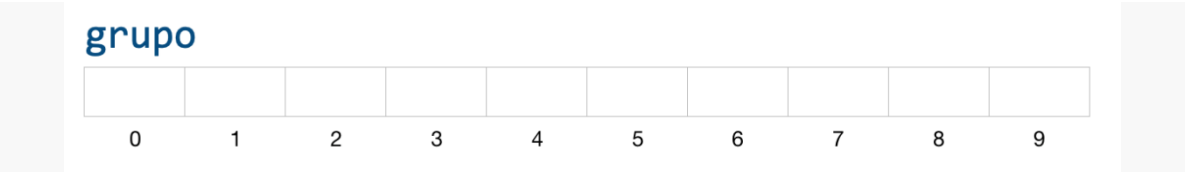
→ **Ciclo for:** Proporciona un bucle que se ejecutará de 0 a n veces pero que incorpora un mecanismo para la inicialización de variables del bucle, controlar la salida o modificar la variable del bucle. Ejemplo for(i=0;i<x;i++){} en donde se toma que x puede ser cualquier valor y la variable de iteración puede incrementar con ++ o decrecer con –



```
for(inicialización;condición de parada;incremento){

        <lista de instrucciones>;

}
```

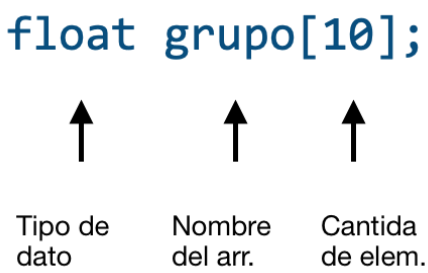
- **Arreglos:**

Un arreglo es una colección de posiciones de almacenamiento de datos, donde cada una tiene el mismo tipo de dato y el mismo nombre. Cada posición de almacenamiento en un arreglo es llamada un elemento del arreglo. (Aitken & Jones, 1994).



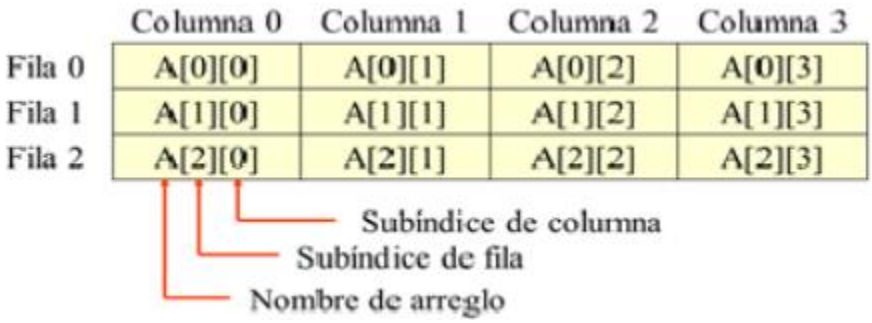
Para declarar o crear un arreglo utilizaremos la siguiente forma:

- Escribe el tipo de dato que almacenará el arreglo
- Escribe el nombre del arreglo
- Entre corchetes, escribe la cantidad de elementos de ese tipo que se almacenarán



Los arreglos en C pueden tener múltiples subíndices. Un uso común de los arreglos con múltiples subíndices es representar tablas de valores, las cuales contienen información organizada en filas y columnas. Para identificar un elemento en especial de una tabla, debemos especificar dos subíndices: el primero identifica la fila del elemento, y el segundo identifica la columna del

elemento. En general, a un arreglo con m filas y n columnas se llama arreglo de m por n, así: m x n.



El lenguaje C, permite al programador declarar matrices de cualquier tipo y prácticamente de cualquier tamaño. En el seudolenguaje, un matriz se declara usando el siguiente formato:

```
<TIPOdeDato><NOMBRE> [<N>][<M>]
```

En este formato aparecen en mayúsculas y entre los caracteres < y > los componentes que el programador puede determinar. Así por ejemplo, si se quiere declarar una matriz con nombre matriz, de dimensión 15x4 y que pueda almacenar datos de tipo carácter, se debe escribir la siguiente línea:

```
char matriz [15][4];
```

Ejemplo de uso de arreglos unidimensionales, bidimensionales y tridimensionales:

```
#include <iostream>



using namespace std;



#define DIM 3 //Definimos un número contante con valor de 3
```

```
const size_t filas = DIM; //Se define la cantidad de filas que habrá con el tipo de dato size_t

const size_t columnas = DIM;

const size_t planos = DIM;

int main()

{

     int*** arreglo = new int**[filas]; //Se hace la declaración del arreglo de tres dimensiones



     int contador = 0;



     for (int x = 0; x < filas; ++x)

     {

               arreglo[x] = new int*[columnas];

               for (int y = 0; y < columnas; ++y)

               {

                         arreglo[x][y] = new int[planos];

                         cout << "**********Plano alojado en: " << "Fila: " << x << " Columna: " << y << " **********" << endl << endl;

                         for (int z = 0; z < planos; ++z)

                         {
```

```
                                arreglo[x][y][z] = contador; //Se asigna el
valor de contador en la posicion x-y-z


                                cout << "Valor " << z << ": " <<
arreglo[x][y][z] << endl; //Se imprimen los valores del plano alojado en la
fila x - columna y


                            }


                            cout << endl <<
"*****************************************************" << endl << endl <<
endl; //Se imprime el separador de la parte inferior


                            contador++; //Se agrega un 1 al contador


                    }


            }


            return 0;


}


/*Nota sobre tipos de datos: se puede observar que en esta ocasión utilizamos
un tipo de dato distinto para asignar el tamaño a nuestro arreglo,


el tipo de dato size_t es utilizado SOLO para asignar tamaños, como es
nuestro caso, se puede interpretar como un tipo entero sin signo,


este tipo de dato NO se puede utilizar para hacer otro tipo de operaciones,
por ejemplo una suma*/




/*Nótese que al declarar el arreglo solo ponemos las filas inicialmente, esto
quiere decir que aún no se asigna un espacio en memoria


para las columnas y lo
```

- **Apuntadores:**

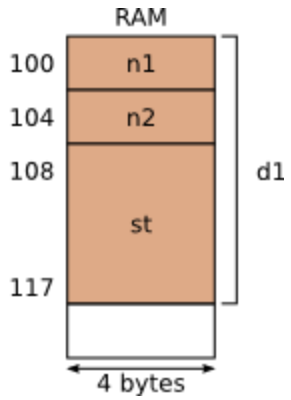
Los punteros son uno de los aspectos más potentes de la programación en C, pero también uno de los más complejos de dominar. Los punteros permiten manipular la memoria del ordenador de forma eficiente. Dos conceptos son fundamentales para comprender el funcionamiento de los punteros:

1. El tamaño de todas variables y su posición en memoria.
2. Todo dato está almacenado a partir de una dirección de memoria. Esta dirección puede ser obtenida y manipulada también como un dato más.

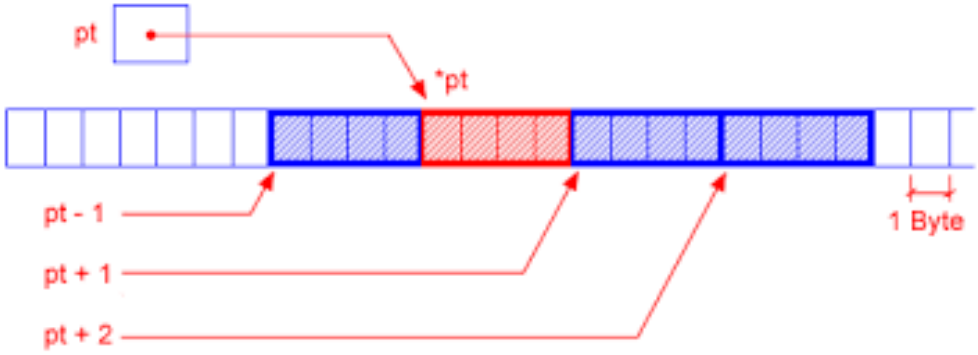
Los punteros son también una de las fuentes de errores más frecuente. Dado que se manipula la memoria directamente y el compilador apenas realiza comprobaciones de tipos, el diseño de programas con punteros requiere una meticulosidad muy elevada que debe ir acompañada de una dosis idéntica de paciencia. Programar eficientemente usando punteros se adquiere tras escribir muchas líneas de código, pero requiere una práctica sostenida.

Todos los datos se almacenan a partir de una dirección de memoria y utilizando tantos bytes como sea necesario. Por ejemplo, considera la siguiente definición de datos y su correspondiente representación en memoria a partir de la dirección 100.

```
struct contact
{
    int n1;
    float n2;
    char st[10];
} d1;
```



Asumiendo que int y float ocupan 4 bytes cada uno, y char ocupa 1 byte, si d1 está almacenada a partir de la posición 100, entonces su campo n1 tiene esa misma dirección, el campo n2 está en la posición 104 y el campo st está almacenado a partir de la posición 108.



En C, al contrario que en otros lenguajes de programacion, se puede obtener directamente la direccion de memoria de cualquier variable. Esto es posible hacerlo con el operador unario "&"; asi:

```
char a;             /* Variable 'a' de tipo char */

printf("la direccion de memoria de 'a' es: %p \n", &a);
```

y para obtener lo apuntado por un puntero se utiliza el operador unario "*" de esta forma:

```
char a;             /* Variable 'a' de tipo char */
char *pchar;        /* Puntero a char 'pchar' */

pchar = &a;         /* 'pchar' <- @ de 'a' */

printf("la direccion de memoria de 'a' es: %p \n", &a);
printf("y su contenido es : %c \n", *pchar);
```

Uno de los casos mas comunes donde se ve la relacion entre estos dos operadores es la declaracion y utilizacion de funciones:

```
void Funcion ( int *int_pointer )
/* Paso de una variable de tipo entero por REFERENCIA */
/* equivalente en Modula 2: PROCEDURE Funcion ( VAR a:INTEGER ) */


int a;
a=6;
```

Funcion (&a); /* ya que la declaracion de la funcion pide la
direccion de una variable de tipo entero */

- **Aritmética de apuntadores:**

Se trata de un conjunto de operaciones que podemos realizar en los punteros que representan un arreglo para manipularlos. Estas operaciones incluyen navegación de los elementos, cómputo de la distancia entre elementos y comparación de las direcciones de los elementos.

Si incrementamos o decrementamos un puntero, éste moverá su dirección en base al tamaño del tipo de dato del puntero. Esto es útil siempre y cuando no salgamos del espacio en la memoria que ocupan los datos.

```
int numeros[5]{11, 22, 33, 44, 55}; // arreglo de numeros
int *p_numeros = numeros;           // puntero al arreglo

std::cout << *(p_numeros + 0) << std::endl; // primer numero
std::cout << *(p_numeros + 1) << std::endl; // segundo numero
std::cout << *(p_numeros + 2) << std::endl; // tercer numero
std::cout << *(p_numeros + 3) << std::endl; // cuarto numero
std::cout << *(p_numeros + 4) << std::endl; // quinto numero
```

Si nos salimos del espacio en la memoria obtendremos resultados inestables y por tanto un mal funcionamiento del programa:

```
std::cout << *(p_numeros + 10) << std::endl; // fuera de rango
```

Esta lógica la podemos usar con el nombre del arreglo refiriéndonos a él como un puntero:

```
std::cout << *(numeros + 0) << std::endl; // primer numero
std::cout << *(numeros + 1) << std::endl; // segundo numero
std::cout << *(numeros + 2) << std::endl; // tercer numero
std::cout << *(numeros + 3) << std::endl; // cuarto numero
std::cout << *(numeros + 4) << std::endl; // quinto numero
```

- **Recursividad:**

En C, las funciones pueden llamarse a sí mismas. Si una expresión en el cuerpo de una función llama a la propia función, se dice que ésta

es *recursiva*. La recursividad es el proceso de definir algo en términos de sí mismo y a veces se llama *definición circular*.

Los ejemplos de recursividad abundan.

Para que en lenguaje de computadora sea recursivo, una función debe ser capaz de llamarse a sí misma. Un sencillo ejemplo es la función fact(), que calcula el factorial de un entero. El factorial de un número N es el producto de todos los número entre 1 y N.

```
1  unsigned long long factorial(unsigned long long numero) {
2      // Si hemos llegado a 1, detenemos la recursión
3      if (numero <= 1)
4          return 1;
5      return numero * factorial(numero - 1); // Restar 1
6  }
```

Gráficos en Lenguaje C

La paquetería de gráficos proviene del lenguaje c++ de manera directa, sin embargo, se puede crear un programa de gráficos de bajo nivel en C, incorporando formas, colores y fuentes de diseño en el programa, para ello es necesario configurar el compilador que se esté utilizando para que permita su uso, en este caso DevC++, para la configuración se siguen los siguientes pasos:

1. Descargar la versión más actual del compilador
2. Descargar los archivos de encabezado de gráficos
3. Extraer el contenido encontrado en el archivo .rar en una carpeta específica
4. Ir a la ubicación donde se encuentra instalado DevC++ y localizarse en la carpeta MinGW64 donde se copian los archivos graphics.h y winbgim.h en la carpeta de inclusión y en la carpeta \MinGW64\x86_64-w64-mingw32\include
5. Se copia el archivo libbgi.a en la carpeta lib y en \MinGW64\x86_64-w64-mingw32\lib
6. Copiar los archivos ConsoleAppGraphics.template, ConsoleApp_cpp_graph.txt y pegarlos dentro de la carpeta de plantillas en la ubicación donde se instaló devc++.
7. Para terminar de configurar, abrir el compilador e ir a la pestaña “parámetros” en el campo “enlazador” y agregar el texto siguiente:
   -lbgi
   -lgdi32
   -lcomdlg32

-luuid
-loleaut32
-lole32

8. Se da clic en aceptar y compilar, al momento de ejecutar los proyectos se abrirán dos ventanas, la ventana de consola y la ventana de gráficos, es importante que para los proyectos con programación en c se cree un archivo con extensión .cpp para que puedan ser compatibles los gráficos y el lenguaje de manera simultánea.
9. Para utilizar los gráficos se manejan las funciones “initgraph” o “initwindow” que inicializan el sistema de gráficos cargando un controlador desde el disco (o validando un controlador registrado) y luego poniendo el sistema en modo gráfico, para cerrarlos se utiliza la función “closegraph”.

Funciones gráficas empleadas

→ setfillstyle: determina el tipo de relleno de las figuras y el color

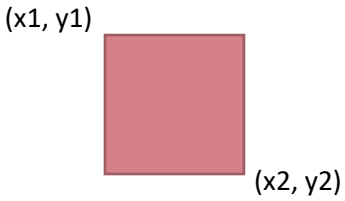
setfillstyle(tipo_de_relleno, color);

→ setbkcolor: cambia el color del fondo de texto

setbkcolor(color);

→ rectangle: se usa para crear figuras de cuatro lados usando dos coordenadas de la siguiente forma:

rectangle(x1, y1, x2, y2);



→ floodfill: rellena un área cerrada con el color y trama actuales

floodfill(x1, y1, color);

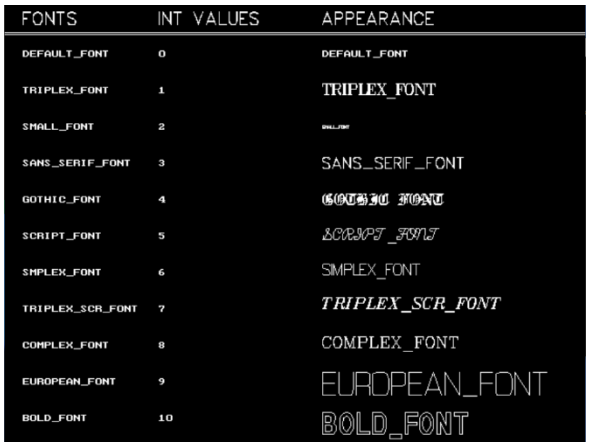
→ outtextxy: define la tipografía y tamaño de letra del texto a escribir

outtextxy(x, y,"char");

→ settextstyle: especifica las características para la salida de texto con fuente

settextstyle(fuente, dirección, tamaño);

Además de ello se consideraron las siguientes tablas de información para establecer la paleta de colores y tipografía de la calculadora:



COLOR	INT VALUES
BLACK	0
BLUE	1
GREEN	2
CYAN	3
RED	4
MAGENTA	5
BROWN	6
LIGHTGRAY	7
DARKGRAY	8
LIGHTBLUE	9
LIGHTGREEN	10
LIGHTCYAN	11
LIGHTRED	12
LIGHTMAGENTA	13
YELLOW	14

Hanoi: Historia

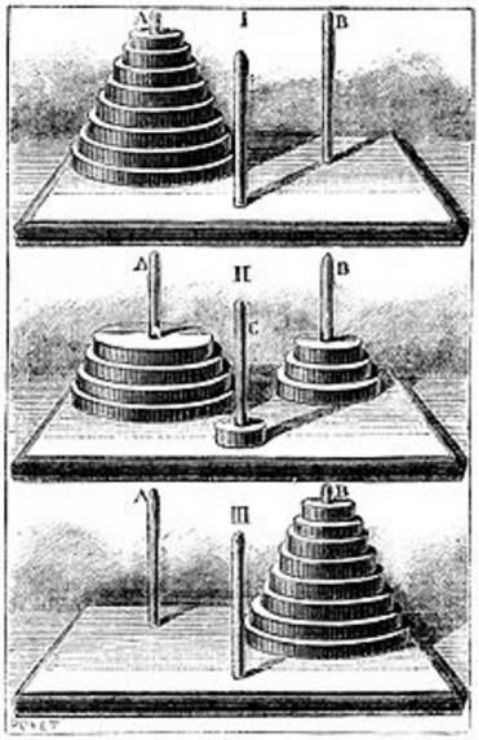
Las torres de Hanói es un rompecabezas matemático inventado en 1883 por el matemático francés Édouard Lucas que consta de tres postes y una serie de discos de diferentes tamaños insertados en uno de los postes. El propósito del rompecabezas es mover todos los discos a uno de los postes vacíos de forma que queden apilados preservando el orden inicial y siguiendo las siguientes reglas:

1. Sólo se puede mover un disco a la vez.
2. Un disco de mayor tamaño no puede estar sobre un disco más pequeño que él mismo.

La solución del problema de las **Torres de Hanói** es muy fácil de hallar, aunque el número de pasos para resolver el problema crece exponencialmente conforme aumenta el número de discos. Cómo ya se ha indicado, el número mínimo de movimientos necesarios para resolver un rompecabezas de la Torre de Hanói es 2 - 1, donde *n* es la cantidad de discos.

Una manera sencilla para saber si es posible terminar el "juego" es que si la cantidad de discos es impar la pieza inicial irá a destino y si es par a auxiliar.

Una forma de resolver el problema se fundamenta en el disco más pequeño, el de más arriba en la varilla de *origen*. En un juego con un número par de discos, el movimiento inicial de la varilla *origen* es hacia la varilla *auxiliar*. El disco n. 2 se debe mover, por regla, a la varilla *destino*. Luego, el disco n. 1 se mueve también a la varilla *destino* para que quede sobre el disco n. 2. A continuación, se mueve el disco que sigue de la varilla *origen*, en este caso el disco n. 3, y se coloca en la varilla *auxiliar*. Finalmente, el disco n. 1 regresa de la varilla *destino* a la *origen* (sin pasar por la *auxiliar*), y así sucesivamente. Es decir, el truco está en el disco más pequeño.



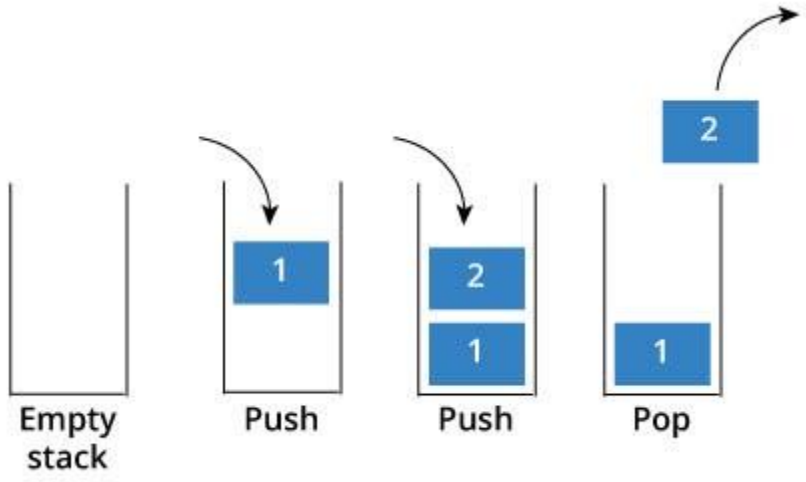
INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

Algoritmo base de Hanoi y funcionamiento en programación

La pila es una estructura de datos lineal que sigue un orden particular. El orden que sigue es el método LIFO (Last-In, First-Out, último en entrar, primero en salir).

Funciones básicas en las pilas:

- **Push:** agrega un elemento a la pila. Si la pila está llena, retorna 0.
- **Pop:** elimina un elemento de la pila. Si la pila está vacía, retorna 0.
- **Empty:** devuelve verdadero si la pila está vacía, de lo contrario retorna 0.
- **Stacktop:** devuelve el elemento superior de la pila, sin eliminarlo.



- **Algoritmo:**

1. si origen == 1 entonces
    1. sacar disco de la torre origen.
    2. insertar disco a torre destino.
2. caso contrario
    1. t_Hanoi(*{1, …, n-1}*, origen, destino, auxiliar) //mover todas los discos menos el más grande (*n*) a la torre auxiliar.
    2. mover disco *n* a torre destino
    3. t_Hanoi(auxiliar, origen, destino) //mover todos los discos restantes, *{1, …, n-1},* encima del disco *n.*
3. Terminar.

- **Código en lenguaje c:**

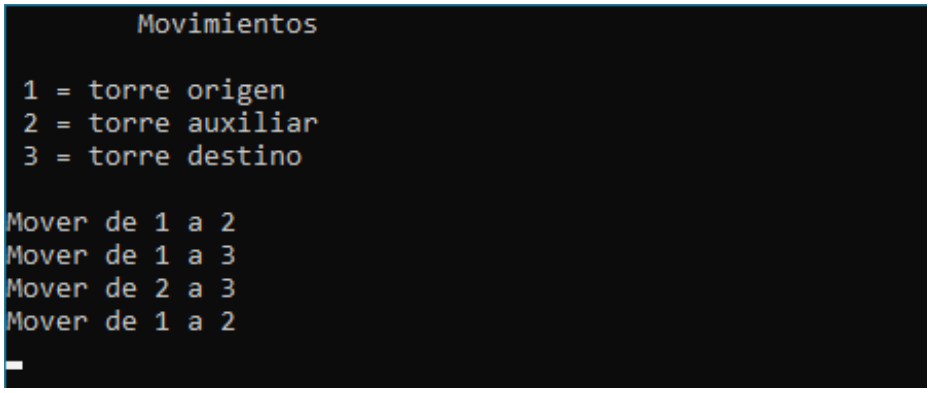
```
void hanoi(int n,int com, int aux, int fin){

  if(n==1){
        printf("%c->%c",com,fin);
  }
  else{
        hanoi(n-1,com,fin,aux);
        printf("\n%c->%c\n",com,fin);
        hanoi(n-1,aux,com,fin);
  }
}
```

Desarrollo: Hanoi en modo gráfico con c

Interfaz

La interfaz del proyecto Hanoi cuenta de dos partes, la primera es en torno a los gráficos usados con c y c++ donde se tienen las tres torres y los discos apilados al inicio en la torre de origen, estos se mueven constantemente conforme se resuelve el hanoi recursivo hasta terminar en la torre de destino.

La segunda parte implica la consola de compilación y ejecución del IDE donde se imprimen los movimientos conforme se van realizando en la pantalla gráfica.



Librerías

```c
#include<stdio.h> //libreria estandar
#include<graphics.h> //librería para usar los gráficos
#include<math.h> //libreria para sumar y restar las coordenadas (animacion de gráficos)
```

- stdio.h: significa “encabezado de entrada-salida estándar” es la biblioteca estándar del lenguaje c que contiene definiciones de macro, contantes y declaraciones de funciones y tipos utilizados para diversas operaciones. Las funciones se dividen en dos categorías: funciones para la manipulación de archivos y para la manipulación de entrada y salida.

- graphics.h: contiene las declaraciones y funciones relacionadas con graficación e inicialización del monitor en modo gráfico, originalmente la función es efectuada en el lenguaje c++ por lo cual la extensión del proyecto es .cpp, sin embargo, la programación es realizada en lenguaje c.

- math.h: es un archivo de cabecera de la librería estándar de c diseñado para operaciones matemáticas básicas, la mayoría de estas incluyen el tipo de dato double, así como en las funciones que hacen uso de ángulos se toma en cuenta los radianes.

Variables globales empleadas en el código

```c
//variables globales
int torres[3][10];
int disco_superior[3];
int mover_origen,mover_destino;
int mover_disco;
int longitud, parte_abajo, parte_arriba;
```

- int torres[3][10]: es un arreglo bidimensional, donde las "filas" son los 10 posibles discos y las "columnas" son las tres torres: origen, auxiliar, destino.

- int disco_superior[3]: es un arreglo unidimensional que representa la parte de arriba (discos) de las tres torres, estos se utilizan para ver cual es el siguiente que se moverá.

- int mover_origen, mover_destino: "Mover origen" mueve el disco de la torre de origen hacia la torre auxiliar o la torre destino, "mover destino" mueve el disco de la torre destino hacia la torre origen o la torre auxiliar, esto dependiendo la cantidad de discos que solicite el usuario y el número de movimientos que serán realizados.

- int mover_disco: es una variable de tipo entero, con ella se obtiene el número de discos que se han movido (1-10).

- int longitud, parte_abajo, parte_arriba: son variables de tipo entero que obtienen las coordenadas necesarias para calcular el tamaño de la pantalla y a su vez realizar las animaciones de movimientos de discos, pero principalmente el dibujo inicial de las tres torres con los discos en la torre origen.

Prototipos de funciones y su uso en el código

```
//prototipos de funciones
void push(int mover_destino, int no_disco);//agrega un disco a la torre
int pop(int mover_origen); //elimina un disco de la torre
void dibujo_principal(); //es la plantilla inicial del hanoi (torres y discos)
void animacion(); //realiza la animación de movimiento de los discos
void hanoi(int n, int origen, int auxiliar, int destino); //hace el proceso del hanoi recursivo
void titulo();
```

- void push (int mover_destino, int no_disco): contiene el código necesario para agregar un disco a una torre, si esta se encuentra llena, retorna en cero.

```
void push(int mover_destino, int no_disco){

    //coloca los discos en la torre
    torres[mover_destino-1][++disco_superior[mover_destino-1]]=no_disco;

}
```

- int pop (int mover_origen): contiene el código necesario para eliminar un disco de la torre (moverlo hacia otra torre), si la torre está vacía retorna en cero.

```
int pop(int mover_origen){

    //toma el disco superior de la torre para moverlo hacia otra, lo "elimina"
    return(torres[mover_origen-1][disco_superior[mover_origen-1]--]);

}
```

- void dibujo_principal(): tiene la plantilla principal del programa, calcula en base a las medidas de la pantalla la posición y tamaño de las torres y discos, así como su color y el tipo de relleno que estos tendrán. Se utilizan las variables j, i para iteración de los ciclos for. Dentro del primer ciclo se crean las torres y dentro del segundo ciclo se crean los discos, sus parámetros utilizan para el color el número de discos, de esta forma se agrega 1 al valor hexadecimal del color y así cambiar el color de cada disco, cosa similar ocurre con el tamaño para que todos sean diferentes. Al final del primer for se manda a llamar a la función “título” en la cual se imprime la palabra “hanoi” en la pantalla de gráficos.

```
void dibujo_principal(){

    int j,i,disco; //i, j son variables para iteración de los ciclos
    cleardevice();

    for(j=1;j<=3;j++){

        //parámetros para pintar las torres
        setfillstyle(BKSLASH_FILL ,WHITE); //tipo de relleno de la figura y color
        bar(j*longitud,parte_arriba,j*longitud+5,parte_abajo);

        //parámetros para dibujar los discos en la torre
        for(i=0;i<=disco_superior[j-1];i++){

            disco=torres[j-1][i];

            /*parámetros para pintar los discos, color de relleno y se
            usa el valor del color que va incrementando para cambiar de acuerdo al número de discos*/
            setfillstyle(SOLID_FILL,1+disco);
            bar(j*longitud-15-disco*5,parte_abajo-(i+1)*10,j*longitud+5+15+disco*5,parte_abajo-i*10);

        }
        titulo(); //imprime el titulo en la pantalla gráfica
    }
}
```

- void animador(): es la función encargada de realizar los movimientos de los discos entre las tres torres, primero se tienen designadas las variables principales con las que se obtendrán las coordenadas en "x” y “y” del disco, además de las variables con las que se realizarán las operaciones aritméticas para poder realizar la impresión de una “animación” o desplazamiento del disco. Primero se obtienen las coordenadas de origen que son multiplicadas por la longitud de la pantalla para crear las nuevas coordenadas a donde será movido el disco, posterior a ello se manda a llamar a la función “dibujo principal” donde se utilizará la plantilla base en la que se tiene que trabajar, ahí se tomará el disco superior de la torre de origen y se moverá hacia una torre de destino, para ello se toman en cuenta los parámetros iniciales que cada disco tiene. Conforme va avanzando la recursividad del código de hanoi se van definiendo las torres de “origen”, “auxiliar” y “destino” que van cambiando constantemente hasta que se logre una resolución total del programa.

```
void animador(){

    //mostrar el movimiento que hace el disco de una torre a otra
    int x,y,dif,sign; //x, y son variables de las coordenadas para mover los discos
    mover_disco=pop(mover_origen);
    x=mover_origen*longitud; //se multiplican las coordenadas de origen por la longitud de la pantalla
    y=parte_abajo-(disco_superior[mover_origen-1]+1)*10;

    //se toma el disco superior de la torre
    for(;y>parte_arriba-20;y-=15){

        dibujo_principal(); //se utiliza el dibujo base (torres y discos a mover)

        /*parámetros para pintar los discos, color de relleno y se
          usa el valor del color que va incrementando para cambiar de acuerdo al número de discos*/
        setfillstyle(SOLID_FILL,1+mover_disco);
        bar(x-15-mover_disco*5,y-10,x+5+15+mover_disco*5,y);
        delay(100);//crea un retardo en la ejecución y cierre del programa

    }
    y=parte_arriba-20;
    dif=mover_destino*longitud-x;
    sign=dif/abs(dif);
```

```
    //se mueve el disco hacia la torre de destino
    for(;abs(x-mover_destino*longitud)>25;x+=sign*15){

        dibujo_principal();//se utiliza el dibujo base (torres y discos a mover)

        /*parámetros para pintar los discos, color de relleno y se
        usa el valor del color que va incrementando para cambiar de acuerdo al número de discos*/
        setfillstyle(SOLID_FILL,1+mover_disco);
        bar(x-15-mover_disco*5,y-10,x+5+15+mover_disco*5,y);
        delay(100);//crea un retardo en la ejecución y cierre del programa

    }
    x=mover_destino*longitud;

    //se coloca el disco en la torre de objetivo
    for(;y<parte_abajo-(disco_superior[mover_destino-1]+1)*10;y+=15){

        dibujo_principal();
        setfillstyle(SOLID_FILL,1+mover_disco);
        bar(x-15-mover_disco*5,y-10,x+5+15+mover_disco*5,y);
        delay(100);//crea un retardo en la ejecución y cierre del programa

    }
```

```
    push(mover_destino,mover_disco);
    dibujo_principal();
}
```

- int hanoi(int n, int origen, int auxiliar, int destino): es la función lógica principal de todo el programa, en base a esta se calculan y realizan todos los movimientos del hanoi, recibe cuatro parámetros: n (número de discos), origen (torre de origen), auxiliar (torre auxiliar) y destino (torre destino), la función utiliza la recursividad para plantear las posiciones que tienen que tener cada uno de los discos para poder traspasarse cumpliendo las reglas del hanoi a la torre que le corresponde al final. Así mismo en esta función se imprimen los movimientos realizados en la consola principal del IDE y se actualizan los valores de origen y destino para realizar las animaciones con la función “animador” y concluir el programa.

```
void hanoi(int n, int origen, int auxiliar, int destino){ //funcion que ejecuta el hanoi

    if(n>=1){

        hanoi(n-1,origen, destino, auxiliar);
        printf("Mover de %i a %i\n", origen, destino);
        dibujo_principal();
        delay(1000);//crea un retardo en la ejecución y cierre del programa
        mover_origen=origen;
        mover_destino=destino;

        //realiza la animación del movimiento
        animador();
        hanoi(n-1,auxiliar, origen, destino);

    }
}
```

- int titulo(): es la función más corta del código, contiene los parámetros y el texto que se imprime en la parte superior de la pantalla gráfica al compilar el programa.

```
void titulo(){ //parámetros para imprimir el título en la pantalla gráfica

    settextstyle(8, 0, 5);//fuente, direccion, tamaño
    outtextxy(130, 100, "H a n o i");//x, y, char

}
```

- int main(): es la función principal, aquí se solicita al usuario el número de discos que deseará que tenga el hanoi, posteriormente se limpia la consola y se prepara para imprimir los movimientos que se realicen, desde aquí se habilitan los gráficos y se delimita el tamaño de la pantalla donde se mostrará el hanoi, en la parte de abajo se inicializan los valores de las tres torres y los valores de los discos que serán colocados en las mismas. Posteriormente se obtienen los valores exactos de la pantalla en coordenadas para poder realizar la graficación de las animaciones y una vez hecho eso se llama a la función hanoi para resolver el programa.

```
int main(){

    int i,gd=DETECT,gm,n;
    printf("Ingrese el numero de discos que desea: \n"); //se ingresa la cantidad de discos que se quiere para el hanoi
    scanf("%d",&n); //lee el número de discos

    system("cls"); //limpia la consola
    printf("\tMovimientos\n\n 1 = torre origen\n 2 = torre auxiliar\n 3 = torre destino\n\n");
    initwindow(500,400); //comienzan a utilizarse los gráficos

    for(i=0;i<3;i++){

        disco_superior[i]=-1; //se inicializan las torres vacías

    }

    for(i=n;i>0;i--){

        push(1,i); //se colocan los discos en la torre origen

    }

    longitud=getmaxx()/4; //obtiene las coordenadas máximas de la pantalla en "x", horizontal
    parte_abajo=getmaxy()-50; //obtiene las coordenadas en "x, y" de la parte de abajo de la pantalla
    parte_arriba=getmaxy()/3+100;
    hanoi(n,1,2,3); //se manda a llamar a hanoi para resolverse (en esta misma se imprimen los movimientos)
    delay(4000); //crea un retardo en la ejecución y cierre del programa
    getch();
}
```

Conclusión

Al culminar el desarrollo de este proyecto se obtienen grandes aprendizajes respecto a temas que son esenciales en la programación pero que sobre todo son la base para poder desarrollar una lógica más fuerte en este ámbito, como son el uso de arreglos, funciones, apuntadores o punteros y la recursividad. Para poder realizar el código de Hanoi se realizó una investigación previa acerca de la historia de este problema que originalmente era un juego para así poder conocer sus reglas y programar de acuerdo con las mismas, además se requirió de un conocimiento sólido acerca del lenguaje c y la compatibilidad con los gráficos del lenguaje c++ para poder realizar animaciones de manera adecuada y que fueran lo más atractivas visualmente. Fueron utilizadas sentencias como los “if-else”, ciclos “while”, ciclos “for”, entre otras para poder crear un funcionamiento correcto del código y obtener una compilación exitosa, además implicó un razonamiento matemático para el uso de la aritmética de apuntadores que nos permitió sumar, restar y multiplicar las coordenadas por variables y constantes preestablecidas por el usuario y por el programador, y así conseguir el movimiento de los discos entre las torres, partiendo de la torre origen hasta la torre destino y utilizando como un apoyo la torre auxiliar.

En conclusión, es importante comprender que con los conocimientos básicos sobre el lenguaje de programación se pueden crear cosas de gran relevancia en la vida e incluso se consigue resolver problemas que aunque parezcan simples conllevan un gran análisis y desarrollo, además con el tiempo que se lleve en el desarrollo del proyecto se adquieren habilidades y se logra una mayor comprensión del lenguaje hasta manejarlo en un nivel más avanzado, todo mediante la práctica, lo cual nos beneficia al momento de querer realizar sistemas de mayor nivel.

Bibliografía

Códigos de clave virtual (Winuser.h) - Win32 apps. (2022, 22 septiembre). Microsoft Learn. Recuperado 1 de septiembre de 2022, de https://learn.microsoft.com/es-es/windows/win32/inputdev/virtual-key-codes?fbclid=IwAR14q0Roqzxtdcyp9o7q-fJFq0ns5_33PkL4Zlt00yA3eRKX2q4T1QNxGHY

GeeksforGeeks. (2017, 6 enero). *Basic Graphic Programming in C++*. Recuperado 1 de septiembre de 2022, de https://www.geeksforgeeks.org/basic-graphic-programming-in-c/?fbclid=IwAR00SKLrTdMTIuvaG-E3CcxPextGp1AlG6Oy3GCkMw3QyG8PMnPXzY4QPfE

GeeksforGeeks. (s. f.). *Computer Graphics*. Recuperado 1 de septiembre de 2022, de https://www.geeksforgeeks.org/computer-graphics-2/?fbclid=IwAR20yFLv2QYoCReIj6ClU2MnlWyzcuuZsX--hWtjzwUDtu4dMJ6vZm7o6Z8

GeeksforGeeks. (2021, 1 diciembre). *settextstyle function in C*. Recuperado 1 de septiembre de 2022, de https://www.geeksforgeeks.org/settextstyle-function-c/

GeeksforGeeks. (2017b, enero 6). *Basic Graphic Programming in C++*. Recuperado 1 de septiembre de 2022, de https://www.geeksforgeeks.org/basic-graphic-programming-in-c/

GeeksforGeeks. (2018, 25 enero). *setfillstyle() and floodfill() in C*. Recuperado 1 de septiembre de 2022, de https://www.geeksforgeeks.org/setfillstyle-floodfill-c/

Finalizar programas en C con exit. Cambiar flujo en bucles: break y continue. Ejemplos resueltos (CU00544F). (s. f.). aprenderaprogramar.com. Recuperado 1 de

septiembre de 2022, de https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=943:finalizar-programas-en-c-con-exit-cambiar-flujo-en-bucles-break-y-continue-ejemplos-resueltos-cu00544f&catid=82&Itemid=210#:~:text=En%20programación%20C%20disponemos%20de,exit%20(-1)%3B

Ambito de variables en C. Globales y locales. Undeclared (first use in this function). Ejemplo código (CU00548F). (s. f.). aprenderaprogramar.com. Recuperado 1 de septiembre de 2022, de https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=948:ambito-de-variables-en-c-globales-y-locales-undeclared-first-use-in-this-function-ejemplo-codigo-cu00548f&catid=82&Itemid=210

Temporizador o crear retardo (delay) en C detener ejecuciÃ³n un tiempo (lenguajeC. (s. f.). Recuperado 1 de septiembre de 2022, de https://aprenderaprogramar.com/foros/index.php?topic=5199.0

Faltan detalles - AprendeAProgramar.com. (s. f.). Recuperado 1 de septiembre de 2022, de https://www.aprendeaprogramar.com/referencia/view.php?f=sprintf

C++ con Clase | Librería ANSI C (stdio/sprintf). (s. f.). Recuperado 3 de septiembre de 2022, de https://conclase.net/c/librerias/stdio/sprintf

¿Cual es el uso del operador % en printf de variables en lenguaje C? (2017, 31 marzo). Stack Overflow en español. Recuperado 3 de septiembre de 2022, de https://es.stackoverflow.com/questions/59590/cual-es-el-uso-del-operador-en-printf-de-variables-en-lenguaje-c

Cómo usar la función memcpy en lenguaje C? (s. f.). Linuxteaching. Recuperado 29 de agosto de 2022, de https://www.linuxteaching.com/article/how_to_use_memcpy_function_in_c_language

Faltan detalles - AprendeAProgramar.com. (s. f.-b). Recuperado 29 de agosto de 2022, de https://www.aprendeaprogramar.com/referencia/view.php?f=strlen

stdio.h - Tipos de datos, Variable, Constante, Trabajo, Ejemplo de uso | KripKit. (s. f.). Recuperado 10 de septiembre de 2022, de https://kripkit.com/stdioh/

Windows.h. (2021, 15 octubre). Recuperado 10 de septiembre de 2022, de https://es.wikipedia.org/wiki/Windows.h

WINDOWINFO (winuser.h) - Win32 apps. (2022, 5 octubre). Microsoft Learn. Recuperado 10 de octubre de 2022, de https://learn.microsoft.com/es-es/windows/win32/api/winuser/ns-winuser-windowinfo

Naps. (2021, 28 noviembre). *Ejemplos explicados de arreglos en lenguaje C - Programación Básica*. Naps Tecnología y educación. https://naps.com.mx/blog/ejemplos-explicados-de-arreglos-en-c/

Arreglos Tridimensionales – C++. (2017, 15 mayo). Blog Programación XXI. https://blogprogramacionxxi.wordpress.com/2017/05/15/arreglos-tridimensionales-c/

UC3M. (s. f.). *Capítulo 5. Los punteros en C*. https://www.it.uc3m.es/pbasanta/asng/course_notes/pointers_es.html

Guzman, H. C. (s. f.). *Aritmética de punteros | Apuntes lenguaje C++ | Hektor Profe*. https://docs.hektorprofe.net/cpp/07-punteros-memoria/05-aritmetica-punteros/

Código de C/Visual C - Torres de Hanoi con función recursiva. (s. f.).

https://www.lawebdelprogramador.com/codigo/C-Visual-C/1260-Torres-de-Hanoi-con-funcion-recursiva.html

C – Factorial recursivo. (2019, 3 diciembre). Parzibyte's blog.

https://parzibyte.me/blog/2019/12/03/c-factorial-recursivo/