UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

</SISTEMA DE GESTIÓN PARA RESTAURANTES>

PROYECTO FINAL

ESTRUCTURA DE DATOS Y ALGORITMOS I

ALUMNO: Cadena Luna Iván Adrián

PROFESOR: Martínez Quintana
Marco Antonio

SEMESTRE: 2021-2

FECHA: 12/08/2021

➢ RESUMEN

El Sistema de Gestión para Restaurantes es un programa básico utilizado en los restaurantes. El objetivo principal es realizar las funciones básicas que un camarero y el administrador o propietario del restaurante pueden hacer. Como el nombre del proyecto sugiere, el proyecto es sobre la facturación, pero también cubre el trabajo de un camarero que toma la orden, ya que el programa mostrará el menú, lo que hará que el cliente seleccione un elemento para ordenar, la cantidad deseada, editar su orden, y recibir la facturación.

La ventaja del programa es que no hay necesidad de contratar a una persona para el mismo sólo se requiere un sistema para ejecutarlo. El cliente puede trabajar en el programa y seleccionar los artículos que quiere pedir y, en consecuencia, puede eliminar los artículos si es necesario. Es fácil de usar, ya que funciona como una calculadora, un utilizado puede simplemente añadir o eliminar algún elemento si es necesario y en consecuencia se generará una nueva factura.

El proyecto es sencillo de entender, y el código fuente se ha presentado de forma comprensible. Se utilizó DevC++ como IDE para una mejor compilación y depuración del código, ya que es un editor moderno y estándar.

En resumen, el programa es fácilmente ejecutable y se puede acceder a él con facilidad. Es un gran software para la futura generación, ya que ahorra tiempo y disminuye el trabajo del propietario del restaurante y del camarero también. También ayudará a los propietarios a manejar a sus clientes de una manera mejor y cómoda.

➢ INTRODUCCIÓN

Dirigir un restaurante ya es bastante ajetreado, así que ¿por qué no facilitar los procesos cotidianos con un sistema que le ayude a aliviar la carga de trabajo?

Los restaurantes tienen que hacer frente a muchos procesos cotidianos. Estos pueden ir desde las asignaciones a los empleados, la gestión de los recursos humanos, el control de la asistencia de los empleados hasta la preparación de las nóminas y el registro de las transacciones y la base de datos.

En el mercado actual, la comida, los restaurantes y su gestión tienen un gran valor, ya que cada día aumenta el número de restaurantes y locales de comida que surgen hoy en día. Se puede considerar como un rápido crecimiento en el campo de los negocios y restaurantes de comida y su sistema de gestión. El sistema de gestión que se aplica a cada restaurante es diferente del otro. Algunos restaurantes pueden ser más grandes y otros más pequeños, pero cada restaurante u hotel requiere un sistema de gestión que se denomina Sistema de Gestión para Restaurantes.

Los sistemas de gestión para restaurantes son tecnologías cruciales que permiten a un establecimiento o a una empresa servir mejor a sus clientes y ayudar a los empleados en las transacciones y controles de alimentos y bebidas.

El sistema de gestión para restaurantes es un programa de base de datos que mantiene un registro de todas las transacciones realizadas en el restaurante a diario. El sistema ayuda a la dirección del restaurante a llevar un registro adecuado de todas las transacciones realizadas y de las que seguirá realizando el restaurante, y a mantener la base de datos del mismo, así como hacer más cómodo al cliente hacer sus pedidos.

Si bien la inversión en elementos como el marketing y la decoración contribuirá en gran medida al crecimiento de un restaurante, la inversión en la tecnología adecuada también desempeña un papel importante, y el software de gestión para restaurantes es uno de los "imprescindibles". de gestión de restaurantes. Todos los restaurantes, ya sean pequeños, mediano o grande, se beneficiará enormemente si cambia los procesos manuales de gestión de restaurantes por otros automatizados o basados en software.

➢ DESARROLLO DEL PROYECTO

- Descripción general del proyecto:

SISTEMA DE GESTIÓN PARA RESTAURANTES

Un sistema de gestión de restaurantes es un tipo de software/terminal de punto de venta (TPV) diseñado específicamente para restaurantes, bares, camiones de comida y otros sectores de la industria alimentaria.

Es una herramienta esencial para cualquier restaurante nuevo.

Beneficios del SGR:

- Seguimiento de las ventas y los pedidos
- Ver estados financieros precisos y en tiempo real
- Acceder a los datos de forma fácil y rápida

Características del sistema de gestión:

- Fácil manejo: El sistema de gestión de restaurantes está diseñado para manejar toda la información primaria necesaria para calcular, por ejemplo, las facturas finales o las ventas totales durante todo el día.
- Interactivo: El objetivo principal del sistema de gestión de restaurantes es llegar a un mayor número de clientes y darles a conocer los artículos existentes y nuevos que ofrecen los restaurantes.

- Lista enlazada como base de datos: Como vamos a utilizar la lista enlazada como base de datos en este sistema de gestión para restaurantes, tendremos todas las ventajas de una lista enlazada, tales como que podemos incrementarla o disminuirla en cualquier momento según nuestro menú, ya que es una estructura de datos dinámica.

- Algoritmo completo o parcial de la solución (según el tamaño del proyecto).

1. Entrar a la base de datos centralizada.
    a. Desplegar opción de sección de gestión.
    b. Desplegar opción de sección de cliente.
    c. Desplegar opción salir del programa.
2. Si se elige la sección de gestión.
    a. Mostrar opción de generar historial de ventas.
        i. Si no hay registros, regresar al menú de sección de gestión.
        ii. Si hay registros, mostrarlos.
    b. Mostrar la opción de añadir nuevos artículos al menú de pedidos.
        i. Ingresar nuevo número de serie del artículo.
            1. Si el artículo ya existe, regresar al menú de sección de gestión.
            2. Si el artículo no existe, registrar nuevo artículo.
            3. Ingresar nombre del nuevo artículo.
            4. Ingresar el precio del nuevo artículo.
    c. Mostrar opción de eliminar nuevos artículos del menú de pedidos.
        i. Ingresar número de serie del artículo a eliminar.
            1. Si el artículo no existe, regresar al menú de sección de gestión.
            2. Si el artículo existe, eliminar el registro del artículo.
    d. Mostrar opción de artículo de mostrar opción de pedidos.
        i. Desplegar lista guardada de artículos.
    e. Mostrar opción de volver al menú principal.
        i. Regresar a la base de datos centralizada.
3. Si se elige la sección de gestión.
    a. Mostrar opción de realizar pedido.
        i. Desplegar lista de artículos guardados de artículos en el menú.
        ii. Ingresar el número de serie del artículo deseado.
            1. Si el número de serie no existe, regresar al menú de sección de clientes.
            2. Si el número de serie existe, guardar la elección.

b. Mostrar opción de ver los artículos ordenados.
    i. Si no hay registros, regresar al menú de sección de clientes.
    ii. Si hay registros, mostrarlos.
c. Mostrar opción de eliminar un artículo pedido
    i. Si no hay registros, regresar al menú de sección de clientes.
    ii. Si hay registros, mostrarlos.
        1. Ingresar el número de serie del artículo a eliminar.
            a. Si el número de serie no está guardado, regresar al menú de sección de pedidos.
            b. Si el número de serie existe y está guardado, eliminar su registro.
d. Mostrar opción de mostrar la factura final.
    i. Si no hay registros, regresar al menú de sección de clientes.
    ii. Si hay registros, mostrarlos.
e. Mostrar opción de volver al menú principal.
    i. Regresar a la base de datos centralizada.

4. Si se elige salir del programa.
    a. Cerrar programa.

- Código fuente comentado del proyecto completo:

```
// En este programa se ha implementado la estructura "Lista Enlazada" para realizar varias tareas.

#include<stdio.h>//Este archivo de cabecera proporciona operaciones básicas de I/O (Input/Output) ó E/S (Entrada/Salida) para el programa
#include<stdlib.h>//Este archivo de cabecera incluye funciones de asignación de memoria, control de procesos, conversiones y otras.
#include<string.h>//Este archivo de cabecera incluye funciones de cadena.

struct nodo//Esta es la estructura que proporciona una plantilla para el nodo de la lista enlazada. Muestra una representación de un artículo de comida en el menú.
// Contiene diferentes atributos del nodo, como el nombre del alimento, la cantidad, el precio, los datos, el puntero al nodo siguiente y el puntero al nodo anterior.
//Aquí, con el fin de realizar varias tareas de gestión/administración, así como de los clientes, utilizando una sola estructura, se crea una sola estructura que provee los artículos del menú, así como la orden del cliente.
{
   char comida[50];
   int cantidad;
```

```
    float precio;
    int datos;
    struct nodo *prev;
    struct nodo *next;
};

//apuntadores estructurales globales que se utilizan en todo el programa para crear la lista enlazada y mantenerla.
struct nodo *head_cli = NULL,*nuevonodo,*cola_cli = NULL;
struct nodo *head_ges = NULL, *cola_ges = NULL;
struct nodo *head_s;

//Esta función imprime las opciones disponibles para que el administrador elija
void menugestion()
{
    printf("\n\t\t\t\t\t\t\t1. Ver las ventas totales\n");
    printf("\t\t\t\t\t\t\t2. Anadir nuevos articulos en el menu de pedidos\n");
    printf("\t\t\t\t\t\t\t3. Eliminar articulos del menu de pedidos\n");
    printf("\t\t\t\t\t\t\t4. Mostrar el menu de pedidos\n");
    printf("\t\t\t\t\t\t\t5. Volver al menu principal\n");
    printf("\t\t\t\t\t\t\t   Ingrese su eleccion ---> ");
}

//Esta función imprime las opciones disponibles para que el cliente elija
void menucliente()
{
    printf("\n\t\t\t\t\t\t\t1. Realice su pedido\n");
    printf("\t\t\t\t\t\t\t2. Ver los articulos que ha pedido\n");
    printf("\t\t\t\t\t\t\t3. Eliminar un articulo del pedido\n");
    printf("\t\t\t\t\t\t\t4. Mostrar la factura final\n");
    printf("\t\t\t\t\t\t\t5. Volver al menu principal \n\n");
    printf("\t\t\t\t\t\t\t   Ingrese su eleccion ---> ");
}

//Esta función crea un nodo para la Lista Enlazada del administrador
struct nodo* creargestion(struct nodo *head,int datos, char comida[25], float precio)
{
    nuevonodo = (struct nodo*)malloc(sizeof(struct nodo));

    nuevonodo->datos = datos;
    nuevonodo->precio = precio;
```

```
    nuevonodo-> cantidad = 0;
    strcpy(nuevonodo->comida,comida);
    nuevonodo->next = NULL;
    nuevonodo->prev = NULL;

    struct nodo *temp = head;

    if(temp==NULL)
        head_ges = cola_ges = nuevonodo;
    else
    {
        while(temp->next!=NULL)
            temp=temp->next;

        temp->next=nuevonodo;
        nuevonodo->prev = cola_ges;
        cola_ges = nuevonodo;
    }

    return head_ges;
}

//Esta función crea un nodo para la Lista Enlazada del cliente
struct nodo* crearcliente(struct nodo *head,int datos,int cantidad)
{
    nuevonodo = (struct nodo*)malloc(sizeof(struct nodo));

    struct nodo *temp1 = head_ges;
    int flag = 0;
    while(temp1!=NULL)
    {
        if(temp1->datos==datos)
        {
            flag = 1;
            break;
        }
        temp1 = temp1->next;
    }

    if(flag==1)
    {
        nuevonodo->datos = datos;
        nuevonodo->precio = cantidad *(temp1->precio);
```

```
        nuevonodo-> cantidad = cantidad;
        strcpy(nuevonodo->comida,temp1->comida);
        nuevonodo->next = NULL;
        nuevonodo->prev = NULL;

        struct nodo *temp = head;

        if(temp==NULL)
            head_cli = cola_cli = nuevonodo;
        else
        {
            while(temp->next!=NULL)
                temp=temp->next;

            temp->next=nuevonodo;
            nuevonodo->prev = cola_cli;
            cola_cli = nuevonodo;
        }


    }
    else
    {
        printf("\n\t\t\t\t\t\t\tiEste articulo no se encuentra en el menu!\n");
    }
    return head_cli;
}

//Esta función muestra la respectiva Lista Enlazada entera cuyo puntero de
cabecera se encuentra vacío
void mostrarlista(struct nodo *head)
{
    struct nodo *temp1 = head;
    if(temp1==NULL)
    {
        printf("\n\t\t\t\t\t\t\tiiLa lista esta vacia!!\n\n");
    }
    else
    {
        printf("\n");
        while(temp1!=NULL)
        {
            if(temp1->cantidad==0)
```

```
            printf("\t\t\t\t\t\t\t\t%d\t%s\t%0.2f\n",temp1->datos,temp1->comida,temp1->precio);
        else
        {
            printf("\t\t\t\t\t\t\t\t%d\t%s\t%d\t%0.2f\n",temp1->datos,temp1->comida,temp1->cantidad,temp1->precio);
        }

        temp1 = temp1->next;
      }
      printf("\n");
   }

}

//Esta función mantiene el valor total de las ventas realizadas mediante el establecimiento de otra Lista Enlazada que mantiene un registro de las ventas totales realizadas a cada cliente representado por cada nodo
struct nodo* ventastotales(int datos,int cantidad)
{
   nuevonodo = (struct nodo*)malloc(sizeof(struct nodo));
   int flag = 0;

   struct nodo *temp1 = head_ges;
   while(temp1->datos!=datos)
   {
      temp1 = temp1->next;
   }

   nuevonodo->datos = datos;
   nuevonodo->precio = cantidad*(temp1->precio);
   nuevonodo-> cantidad = cantidad;
   strcpy(nuevonodo->comida,temp1->comida);
   nuevonodo->next = NULL;
   nuevonodo->prev = NULL;

   struct nodo *temp = head_s;

   if(temp==NULL)
      head_s = nuevonodo;
   else
   {
      while(temp->next!=NULL)
```

```
        {
            if(temp->datos==datos)
            {
                flag = 1;
                break;
            }
            temp=temp->next;
        }

        if(flag==1)
        {
            temp->cantidad += nuevonodo-> cantidad;
            temp->precio += nuevonodo->precio;
        }
        else
        {
            temp->next=nuevonodo;
        }
    }

    return head_s;
}

//Esta función realiza la tarea de calcular las ventas totales de cada cliente
void calcularventastotales()
{
    struct nodo *temp = head_cli;
    while(temp!=NULL)
    {
        head_s = ventastotales(temp->datos, temp->cantidad);
        temp=temp->next;
    }
}

//Esta función realiza la tarea de borrar los datos de la Lista Enlazada cuyo
puntero de cabecera se encuentra vacío
//Aquí, los datos que se van a borrar se pasan como parámetro.
struct nodo* eliminar(int datos,struct nodo *head, struct nodo* cola)
{
    if(head==NULL)
    {
        printf("\n\t\t\t\t\t\t\tLa lista esta vacia\n");
    }
```

```
    else
    {
        struct nodo* temp;
        if(datos==head->datos)
        {
            temp = head;
            head = head->next;
            if (head != NULL)
                head->prev = NULL;
            free(temp);
        }
        else if(datos==cola->datos)
        {
            temp = cola;
            cola = cola->prev;
            cola->next = NULL;
            free(temp);
        }
        else
        {
            temp = head;
            while(datos!=temp->datos)
            {
                temp = temp->next;
            }
            (temp->prev)->next = temp->next;
            (temp->next)->prev = temp->prev;
            free(temp);
        }
    }
    return head;
}

//Esta función realiza la tarea de borrar los alimentos de la lista vinculada del
administrador.
int eliminargestion()
{
    printf("\n\t\t\t\t\tIngrese el numero de serie del articulo de comida que desea
eliminar: ");
    int num;
    scanf("%d",&num);

    struct nodo* temp=head_ges;
```

```
    while(temp!=NULL)
    {
        if (temp->datos == num)
        {
            head_ges = eliminar(num, head_ges, cola_ges);
            return 1;
        }
        temp=temp->next;
    }

    return 0;
}

//Esta función realiza la tarea de eliminar los alimentos de la lista vinculada del cliente, es decir, los alimentos pedidos por el cliente.
int eliminarcliente()
{
    printf("\n\t\t\t\t\tIngrese el numero de serie del articulo de comida que desea eliminar: ");
    int num;
    scanf("%d",&num);

    struct nodo* temp=head_cli;
    while(temp!=NULL)
    {
        if (temp->datos == num)
        {
            head_cli = eliminar(num, head_cli, cola_cli);
            return 1;
        }
        temp=temp->next;
    }

    return 0;
}

//Esta función muestra la factura total de los alimentos pedidos por el cliente.
void mostrarfactura()
{
    mostrarlista(head_cli);
    struct nodo *temp = head_cli;
    float total_precio = 0;
    while (temp!=NULL)
```

```
    {
        total_precio +=temp->precio;
        temp = temp->next;
    }

    printf("\t\t\t\t\t\t\t\tPrecio total: %0.02f\n",total_precio);

}

//Esta función realiza la tarea de borrar toda la Lista Enlazada.
struct nodo* eliminarlista(struct nodo* head)
{
    if(head==NULL)
    {
        return NULL;
    }
    else
    {
        struct nodo* temp = head;
        while(temp->next!=0)
        {
            temp = temp->next;
            free(temp->prev);
        }
        free(temp);
        head = NULL;
    }

    return head;
}

//Esta función abre la sección de gestión y proporciona su interfaz y
funcionalidades
void gestion()
{
    printf("\n\t\t\t\t\t   ----------------------------------------------\n");
    printf("\t\t\t\t\t\t   SECCION DE GESTION\n");
    printf("\t\t\t\t\t   ----------------------------------------------\n");
    while(1)
    {
        menugestion(); //imprime la lista de funciones para el administrador

        int opt;
```

```
        scanf("%d",&opt); //escanea la elección del usuario

        if(opt==5)
           break;

        switch (opt)//bloque switch-case que se ejecuta según la opción seleccionada por el usuario
        {
           case 1:
              mostrarlista(head_s);
              break;
           case 2:

              printf("\n\t\t\t\t\t\t\tIngrese el numero de serie del articulo de comida: ");
              int num,flag = 0;
              char nombre[50];
              float precio;
              scanf("%d",&num);

              struct nodo *temp = head_ges;

              while(temp!=NULL)
              {
                 if(temp->datos==num)
                 {
                    printf("\n\t\t\t\t\t\tiiEl articulo de comida con el numero de serie ingresado ya existe!!\n\n");
                    flag = 1;
                    break;
                 }
                 temp = temp->next;
              }

              if(flag==1)
                 break;

              printf("\t\t\t\t\t\t\tIntroduzca el nombre del articulo de comida: ");
              scanf("%s",nombre);
              printf("\t\t\t\t\t\t\tIngrese el precio: ");
              scanf("%f",&precio);
              head_ges = creargestion(head_ges, num, nombre, precio);
              printf("\n\t\t\t\t\t\t\tiiNueva comida anadida a la lista!!\n\n");
```

```
            break;
        case 3:
            if(eliminargestion())
            {
                printf("\n\t\t\t\t\t\t\t### Lista del menu de comida actualizada ###\n");
                mostrarlista(head_ges);
            }
            else
                printf("\n\t\t\t\t\t\t\tiEl producto de comida con el numero de serie ingresado no existe!\n\n");

            break;
        case 4:
            printf("\n\t\t\t\t\t\t\t   ### Menu de pedidos ###\n");
            mostrarlista(head_ges);
            break;

        default:
            printf("\n\t\t\t\t\t\t\tiiEntrada incorrecta!! Por favor, elija una opcion valida\n");
            break;
      }
   }
}

//Esta función abre la sección de clientes y proporciona su interfaz y funcionalidades
void cliente()
{
   int flag=0,j=1;
   char ch;
   printf("\n\t\t\t\t\t   ----------------------------------------------\n");
   printf("\t\t\t\t\t\t\t   SECCION DE CLIENTES\n");
   printf("\t\t\t\t\t   ----------------------------------------------\n");
   while(1)
   {
      menucliente();//imprime la lista de funciones para el cliente

      int opt;
      scanf("%d",&opt);//escanea la elección del usuario

      if(opt==5)
```

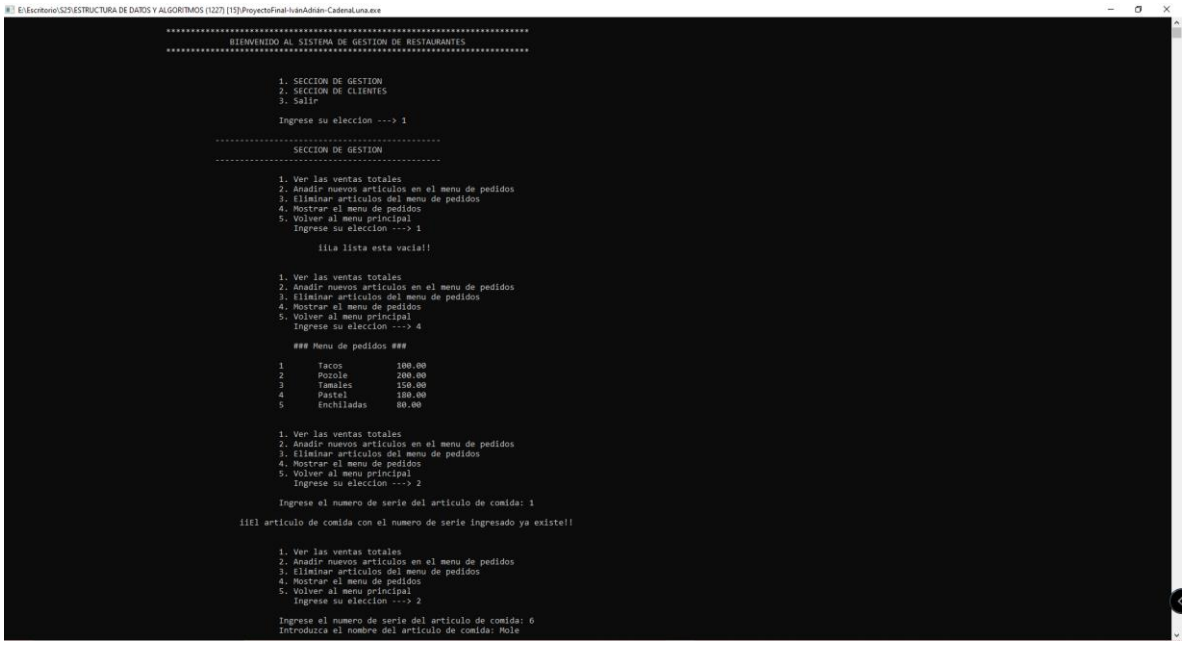
```
            break;

                //bloque switch-case que se ejecuta según la opción seleccionada por el usuario
        switch (opt)
        {
            case 1:
                mostrarlista(head_ges);
                printf("\n\t\t\t\t\t\t\tIngrese el numero correspondiente al articulo que desea ordenar: ");
                int n;
                scanf("%d",&n);
                printf("\t\t\t\t\t\t\tIngrese la cantidad: ");
                int cantidad;
                scanf("%d",&cantidad);
                head_cli = crearcliente(head_cli, n, cantidad);
                break;
            case 2:
                printf("\n\t\t\t\t\t\t\t  ### Lista de articulos ordenados ###\n");
                mostrarlista(head_cli);
                break;
            case 3:
                if(eliminarcliente())
                {
                    printf("\n\t\t\t\t\t\t\t### Lista de articulos ordenados actualizada ###\n");
                    mostrarlista(head_cli);
                }
                else
                    printf("\n\t\t\t\t\t\t\tiiEl articulo de comida con el numero de serie ingresado no existe!!\n");
                break;
            case 4:
                calcularventastotales();
                printf("\n\t\t\t\t\t\t\t  ### Factura final ###\n");
                mostrarfactura();
                head_cli = eliminarlista(head_cli);
                printf("\n\t\t\t\t\t\t\tPulse cualquier tecla para volver al menu principal:\n\t\t\t\t\t\t\t");
                fflush(stdin);
                ch=fgetc(stdin);
                flag=1;
                break;
```

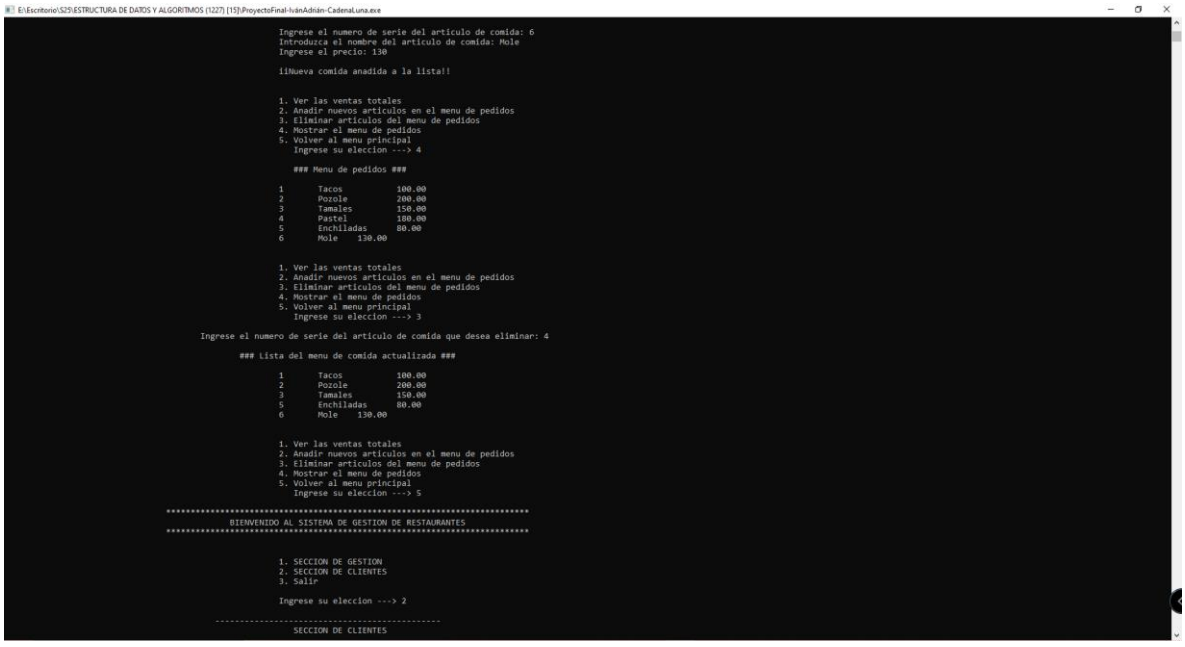
```
            default:
                printf("\n\t\t\t\t\t\tiiEntrada incorrecta!! Por favor, elija una opcion
valida\n");
                break;
        }
        if(flag==1)
            break;
    }
}

//Esta función imprime la interfaz de bienvenida y abre el menú principal
donde se puede seleccionar la opción a la que se quiere ir.
void mainnenu()
{
    printf("\n
*********************************************************************************\n");
    printf("                                          BIENVENIDO AL SISTEMA DE
GESTION DE RESTAURANTES\n");
    printf("
*********************************************************************************\n\n\n");
    printf("\t\t\t\t\t\t\t1. SECCION DE GESTION \n");
    printf("\t\t\t\t\t\t\t2. SECCION DE CLIENTES \n");
    printf("\t\t\t\t\t\t\t3. Salir \n\n");
    printf("\t\t\t\t\t\t\tIngrese su eleccion ---> ");
}

int main()//A partir de aquí comienza la ejecución real del programa
{
        //Aquí hemos inicializado la Lista Vinculada del administrador, es decir,
el Menú de Comida con 5 artículos
    head_ges = creargestion(head_ges,1,"Tacos",100);
    head_ges = creargestion(head_ges,2,"Pozole",200);
    head_ges = creargestion(head_ges,3,"Tamales",150);
    head_ges = creargestion(head_ges,4,"Pastel",180);
    head_ges = creargestion(head_ges,5,"Enchiladas",80);

    while(1)
    {
        mainnenu();
        int eleccion;
        scanf("%d",&eleccion);//escanea la elección del usuario
```

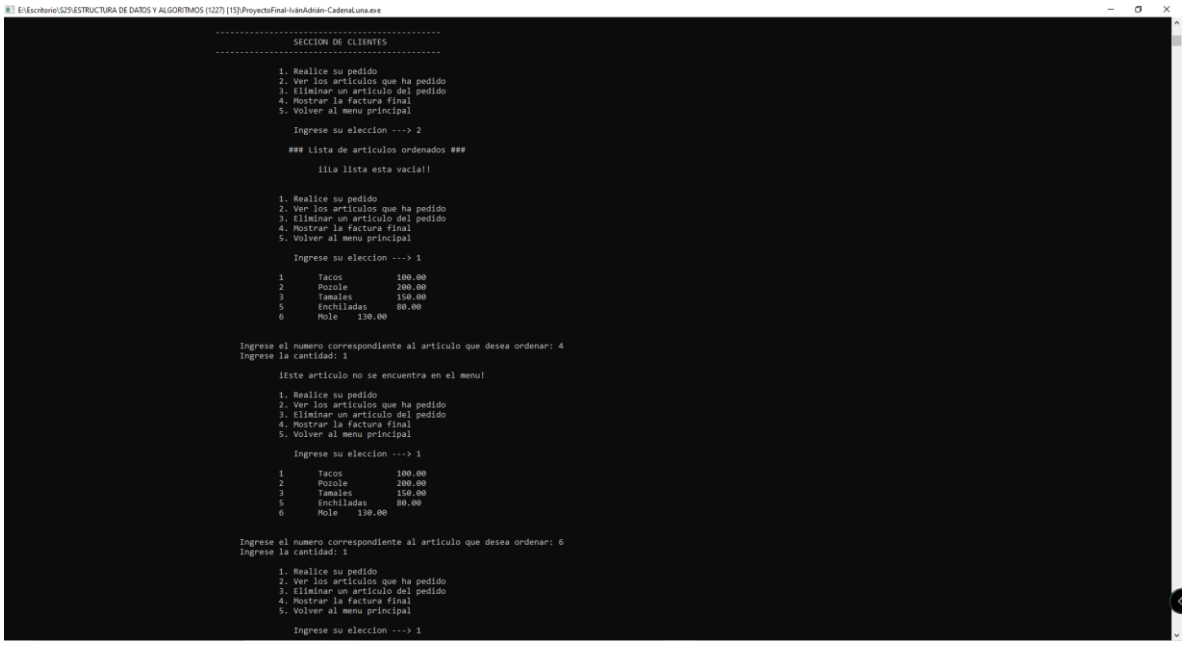
```
        if(eleccion==3)
        {
            printf("\n\n\t\t\t\t\t\t\t**********iiGracias!!**********\n");
            break;
        }

                //bloque switch-case que se ejecuta según la opción seleccionada por el usuario
        switch (eleccion)
        {
            case 1:
                gestion();
                break;
            case 2:
                cliente();
                break;
            case 3:
                break;

            default:
                printf("\n\t\t\t\t\t\tiiEntrada incorrecta!! Por favor, elija una opcion valida\n");
                break;
        }
    }
}
```

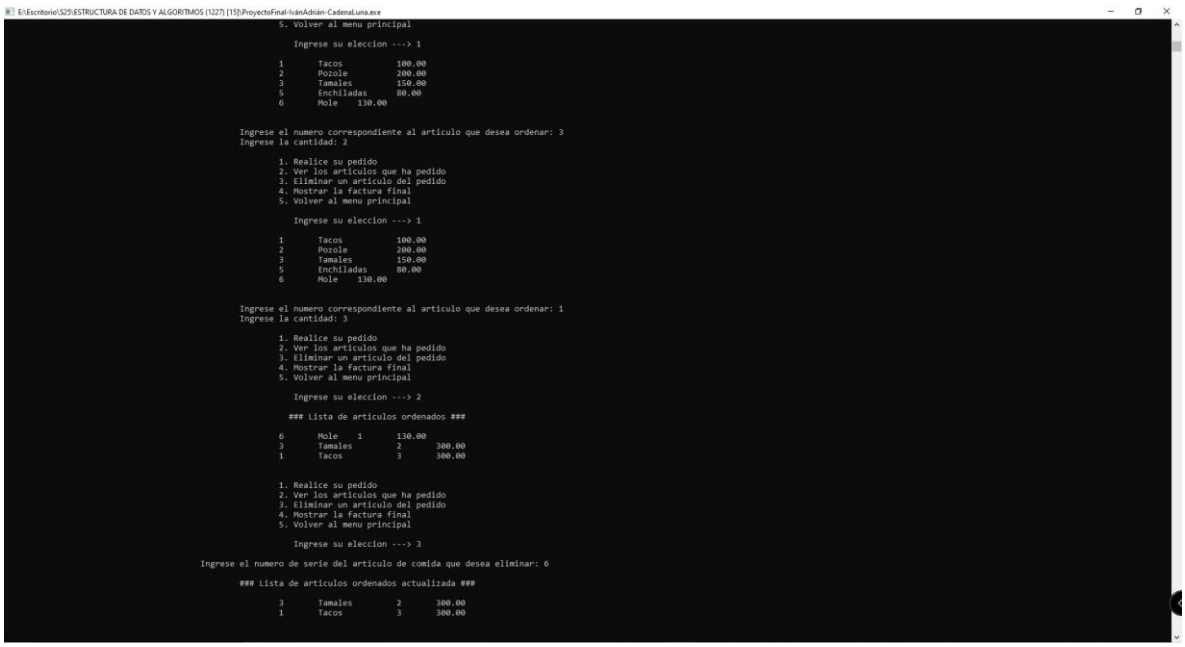
➢ RESULTADOS DEL PROYECTO

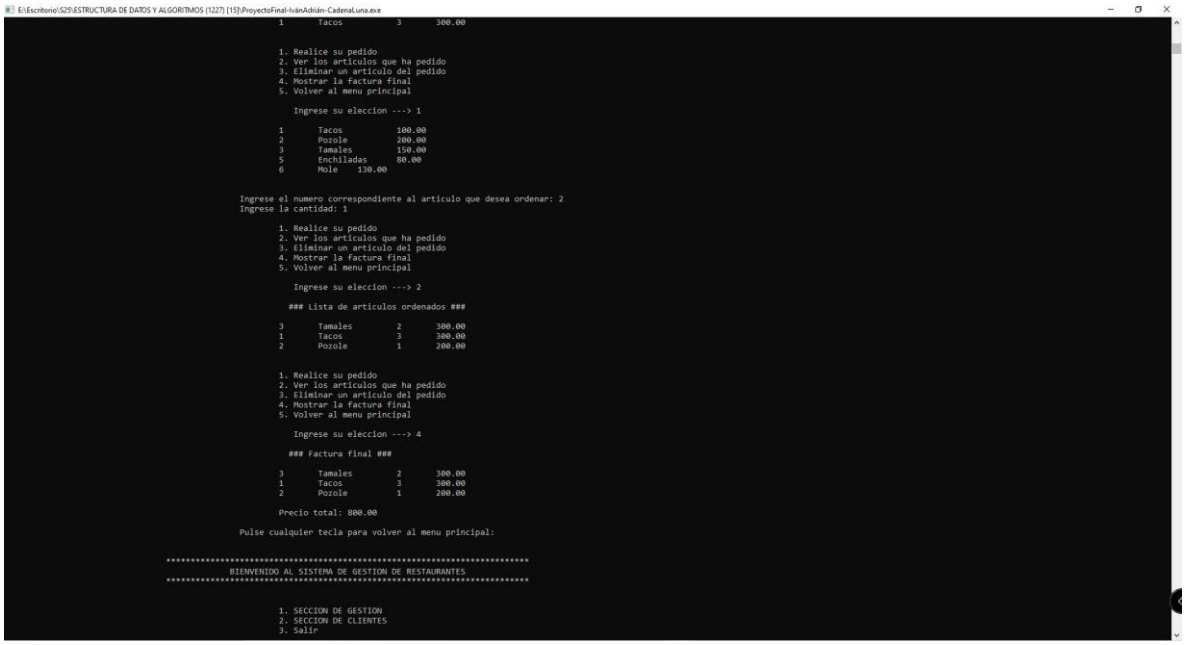
- Capturas de pantalla del funcionamiento de su proyecto

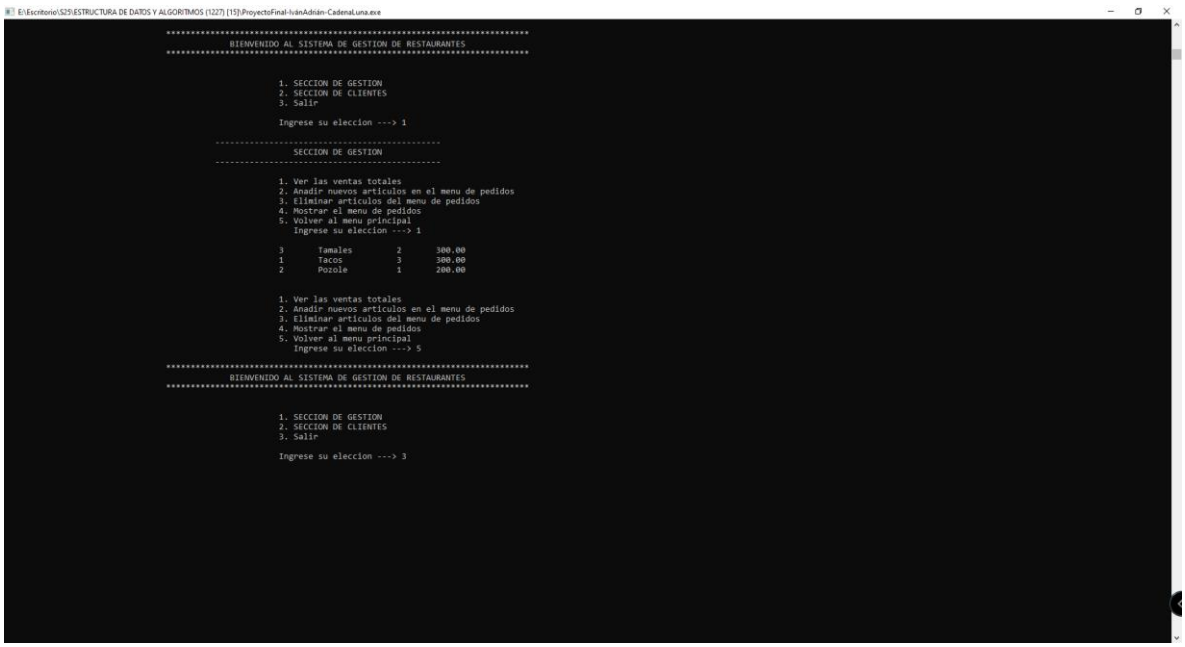





E:\Escritorio\S2S\ESTRUCTURA DE DATOS Y ALGORITMOS (1227) [15]\ProyectoFinal-IvánAdrián-CadenaLuna.exe
SECCION DE CLIENTES
1. Realice su pedido
2. Ver los articulos que ha pedido
3. Eliminar un articulo del pedido
4. Mostrar la factura final
5. Volver al menu principal
Ingrese su eleccion ---> 2
Lista de articulos ordenados
¡¡La lista esta vacia!!
1. Realice su pedido
2. Ver los articulos que ha pedido
3. Eliminar un articulo del pedido
4. Mostrar la factura final
5. Volver al menu principal
Ingrese su eleccion ---> 1
1 Tacos 100.00
2 Pozole 200.00
3 Tamales 150.00
5 Enchiladas 80.00
6 Mole 130.00
Ingrese el numero correspondiente al articulo que desea ordenar: 4
Ingrese la cantidad: 1
¡Este articulo no se encuentra en el menu!
1. Realice su pedido
2. Ver los articulos que ha pedido
3. Eliminar un articulo del pedido
4. Mostrar la factura final
5. Volver al menu principal
Ingrese su eleccion ---> 1
1 Tacos 100.00
2 Pozole 200.00
3 Tamales 150.00
5 Enchiladas 80.00
6 Mole 130.00
Ingrese el numero correspondiente al articulo que desea ordenar: 6
Ingrese la cantidad: 1
1. Realice su pedido
2. Ver los articulos que ha pedido
3. Eliminar un articulo del pedido
4. Mostrar la factura final
5. Volver al menu principal
Ingrese su eleccion ---> 1


E:\Escritorio\S2S\ESTRUCTURA DE DATOS Y ALGORITMOS (1227) [15]\ProyectoFinal-IvánAdrián-CadenaLuna.exe
5. Volver al menu principal
Ingrese su eleccion ---> 1
1 Tacos 100.00
2 Pozole 200.00
3 Tamales 150.00
5 Enchiladas 80.00
6 Mole 130.00
Ingrese el numero correspondiente al articulo que desea ordenar: 3
Ingrese la cantidad: 2
1. Realice su pedido
2. Ver los articulos que ha pedido
3. Eliminar un articulo del pedido
4. Mostrar la factura final
5. Volver al menu principal
Ingrese su eleccion ---> 1
1 Tacos 100.00
2 Pozole 200.00
3 Tamales 150.00
5 Enchiladas 80.00
6 Mole 130.00
Ingrese el numero correspondiente al articulo que desea ordenar: 1
Ingrese la cantidad: 3
1. Realice su pedido
2. Ver los articulos que ha pedido
3. Eliminar un articulo del pedido
4. Mostrar la factura final
5. Volver al menu principal
Ingrese su eleccion ---> 2
Lista de articulos ordenados
6 Mole 1 130.00
3 Tamales 2 300.00
1 Tacos 3 300.00
1. Realice su pedido
2. Ver los articulos que ha pedido
3. Eliminar un articulo del pedido
4. Mostrar la factura final
5. Volver al menu principal
Ingrese su eleccion ---> 3
Ingrese el numero de serie del articulo de comida que desea eliminar: 6
Lista de articulos ordenados actualizada
3 Tamales 2 300.00
1 Tacos 3 300.00





- Tabla de recursos informáticos [software y hardware] necesarios para llevar a cabo el proyecto

Software	Hardware
Windows 10 Dev-C++ Visual Studio Core	Laptop Lenovo Ideapad 330-14AST AMD A4 4GB RAM 1TB DD

- Tabla de costos propuestos para el desarrollo del proyecto

Tipo de software	Aplicación para negocios
Tiempo invertido en el desarrollo	48 horas
Cálculo en base a un sueldo laboral	$125/hora
Costos (equipo, licencias, base de datos, certificados, dominio, etc.)	$500
TOTAL	$6500

- Diagrama de Gantt para la elaboración del proyecto (6-Agosto-2021 al 13Agosto-2021 o periodo utilizado)

Etapa	**06/08/21**	**08/08/21**	**10/08/21**	**11/08/21**	**12/08/21**
Idea del proyecto	**X**				
Investigación		**X**			
Algoritmo		**X**			
Listas Enlazadas Simples			**X**	**X**	**X**
Listas Enlazadas Dobles			**X**	**X**	**X**
Funciones Gestión			**X**		
Funciones Cliente			**X**		
Menú Gestión				**X**	
Menú Cliente				**X**	
Cálculos Precios				**X**	**X**
Solución de errores					**X**

- Canal de YouTube donde se encuentra el video: https://youtu.be/FZLyghDxYik

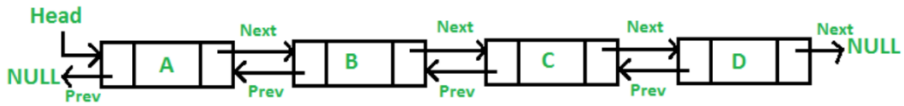
- Repositorio de GitHub del Proyecto Final: https://github.com/CaliRed7/EDA-I/tree/main/05%20Proyecto%20Final

➢ CONCLUSIONES

Este proyecto se desarrolló mediante el uso de listas, un tipo de estructura de datos lineal y dinámica; las cuales permitieron buscar, insertar y eliminar los nodos a los que se atribuyeron los datos de los artículos de comida, el nombre del alimento, la cantidad a ordenar, su precio, los datos y los elementos sucesores y predecesores.

La importancia de la materia radica en el uso de esta estructura de datos; para explicarlo más detalladamente haré una comparación entre un Array vs una Lista Enlazada, que, aunque bien se sabe que ambos se utilizan para almacenar datos lineales de tipos similares, hay ciertas ventajas de usar una sobre la otra.
En el caso de las listas enlazadas tenemos un tipo de estructura no primitivo, la opción de establecer un tamaño dinámico, la herramienta de inserción y de borrado, además del uso de índices.
Además, es muy importante mencionar que cuanto más avanzadas son este tipo de estructuras, más posibilidades tenemos, en este caso con el uso de listas enlazadas dobles y no únicamente simples.
A continuación, una breve representación de las listas enlazadas dobles como se usaron en el desarrollo de este proyecto:

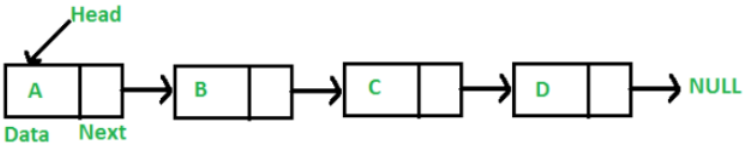
```
struct Nodo {
int datos;
struct Nodo* next;
struct Nodo* prev;
};
```



Ventajas sobre las listas enlazadas simples:

- Transversales: Una lista enlazada doble puede ser recorrida tanto hacia adelante como hacia atrás.
- Inserción: Podemos insertar rápidamente un nuevo nodo antes de un nodo dado.
- Eliminación: La operación de borrado en una lista enlazada doble es más eficiente si se da el puntero al nodo que se va a borrar.

Hablando técnicamente, la implementación de este tipo de estructuras es simplemente magnifica, pues gracias a ser estructuras de datos lineales, las listas enlazadas tienen aplicaciones en diferentes campos de la industria tecnológica.
Una lista enlazada es una estructura de datos lineal, en la que los elementos no se almacenan en ubicaciones de memoria contiguas. Los elementos de una lista enlazada se enlazan mediante punteros como se muestra en la imagen:



Gracias a estas características, las listas enlazadas tienen muchas aplicaciones en el mundo real, las cuales se encuentran presentes en nuestra vida diaria como:

- Visores de imágenes.
- Página anterior y siguiente en navegador web.
- Reproductores de música.

Incluso, si seguimos pensamos en grande, al implementar listas enlazadas circulares, las posibilidades se multiplican:

- Implementación de colas.
- Múltiples aplicaciones que se ejecutan en un PC.
- Estructuras de datos avanzadas.

Aplicaciones en ciencias de la computación:

- Implementación de pilas y colas.
- Implementación de grafos: La representación de listas de adyacencia de los gráficos es la más popular, que utiliza una lista enlazada para almacenar vértices adyacentes.
- Asignación dinámica de memoria: Utilizamos listas enlazadas de bloques libres.
- Mantenimiento de un directorio de nombres.
- Realización de operaciones aritméticas con enteros grandes.
- Manipulación de polinomios almacenando constantes en el nodo de la lista enlazada.
- Representación de matrices dispersas.

Personalmente, con el desarrollo de este proyecto y con los aprendizajes obtenidos de la materia, conceptualice mejor los tipos de datos complejos, así como su construcción a partir de datos simples y sus características principales para su aplicación en la solución de un problema específico, en este caso un programa para la gestión de un restaurante, pero teniendo presente la infinita cantidad de casos y aplicaciones en las que el uso de las estructuras de datos (arreglos, pilas, colas, listas) y algoritmos pueden hacer una enorme diferencia. y en un menor tiempo

Saber cómo manejar nuestros datos es de gran importancia, sobre todo cuando manejamos grandes volúmenes de información. además, aprender sobre estructuras de datos hizo que la forma en la que veía los problemas cambiara radicalmente, pues ya no se trató sobre datos genéricos, ahora los

problemas de programación fueron atacados viéndolos como listas, y tal vez en el futuro cercano, también como grafos o árboles. Nuestro nivel de abstracción aumentara y podremos programar problemas complejos en un menor tiempo. además de que tendremos un código más eficiente y de mayor calidad.

➢ REFERENCIAS

- Introduction to Algorithms. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, McGraw-Hill.
- The Algorithm Design Manual. Steven S. Skiena, Springer.
- Solano, J. (2017). *Manual de prácticas del laboratorio de Estructuras de datos y algoritmos I*. Recuperado de http://odin.fi-b.unam.mx/salac/practicasEDAA/MADO-19_EDAI.pdf
- Sáenz, E. (2017). *Manual de prácticas del laboratorio de Estructuras de datos y algoritmos II*. Recuperado de http://odin.fi-b.unam.mx/salac/manualEDyA2.php
- Arrieta, E. (s.f.). *Manual de Estructura de Datos*. Recuperado de https://www.itsa.edu.co/docs/25-E-Arrieta-Manual-de-Estructura-de-Datos.pdf
- *Introduction to Linked Lists*. Recuperado de https://www.techiedelight.com/introduction-linked-lists/
- *Linked List Implementation in C*. Recuperado de https://www.techiedelight.com/linked-list-implementation-part-1/
- *Linked List – Insertion at Tail | C, Java, and Python Implementation*. Recuperado de https://www.techiedelight.com/linked-list-implementation-part-2/
- *Static Linked List – C, Java, and Python*. Recuperado de https://www.techiedelight.com/static-linked-list-c/
- *Clone a Linked List*. Recuperado de https://www.techiedelight.com/clone-given-linked-list/
- *Delete a linked list in C*. Recuperado de https://www.techiedelight.com/delete-linked-list/
- *Estructuras de datos – 2. Listas enlazadas: Teoría*. makigas: tutoriales de programación. (6 oct 2015). https://www.youtube.com/watch?v=0NzAFk1CwaQ
- *Estructuras de datos – 3. Listas enlazadas en C (COMPLETO)*. makigas: tutoriales de programación. (8 oct 2015). https://www.youtube.com/watch?v=vldM-3PYAmo
- *Estructura de datos en C - Programación Explicada.* Tutoriales de Programación Explicada. (14 may 2019).

https://www.youtube.com/playlist?list=PLsaihF7BSsEmisgClhlgeFME2UBwUqnWz
- *Glosario informático.* https://www.glosarioit.com

➢ GLOSARIO DE TÉRMINOS UTILIZADOS

- **Arreglo**: Es una estructura de datos en la cual estos datos son del mismo tipo y se caracteriza por almacenar sus elementos en posiciones continuas de memoria, tienen un único nombre que representa a todos los elementos, los cuales a su vez se diferencian mediante un índice o subíndice. Los elementos pueden ser accedidos de forma directa.
- **Colección**: Las colecciones permiten trabajar con un conjunto de variables u objetos relacionados, y acceder a cada uno mediante un índice o clave única que lo identifica.
- **Estructura**: Las estructuras o registros permiten agrupar un conjunto de variables bajo un mismo nombre. Cada una de estas variables serán los campos de la estructura y para invocarlos se usará el nombre de la estructura, antecediendo al nombre de la variable. Estas variables pueden ser de distinto tipo y se suelen agrupar por tener una finalidad en común.
- **Estructura de datos**: Es una colección o un conjunto de datos organizados de alguna manera en particular. Pueden ser de dos tipos: estáticos (cantidad fija de memoria cuando se declara la variable) y dinámicos (pueden crecer o decrecer cuando el programa se está ejecutando).
- **Dato**: Factores tales como, caracteres o símbolos almacenados por un computador, de los que se deriva información después de su procesamiento. Los datos, en consecuencia, incluyen algo más que números; abarcan toda la información que se ingresa en la computadora. La materia prima que se procesa o maneja por los programas son los datos.
- **Matriz**: Una matriz es un conjunto de elementos de un mismo tipo, a los que se accede individualmente mediante un índice. El uso de matrices nos permite manipular un gran número de datos relacionados sin necesidad de definir una variable para cada uno de ellos. Cada uno de los elementos de una matriz puede ser de cualquiera de los tipos de datos que se conocen, escalares o no. Independientemente de esto, el conjunto de elementos que forma la matriz es uno de los tipos estructurados. Estos arrays (arreglos) poseen más de una dimensión.
- **Nodo**: Es cada uno de los elementos de una lista enlazada, un árbol o un grafo en una estructura de datos. Los elementos de que constan las estructuras no lineales se conocen con el nombre de nodos. Cada nodo tiene sus propias características y cuenta con varios campos; al menos uno de éstos debe funcionar como punto de referencia para otro nodo.

- **Listas simples enlazadas**: Es una lista enlazada de nodos, donde cada nodo tiene un único campo de enlace. Una variable de referencia contiene una referencia al primer nodo, cada nodo (excepto el último) enlaza con el nodo siguiente, y el enlace del último nodo contiene NULL para indicar el final de la lista. Aunque normalmente a la variable de referencia se la suele llamar top, se le podría llamar como se desee.
- **Listas doblemente enlazadas**: Un tipo de lista enlazada más sofisticado es la lista doblemente enlazada o lista enlazadas de dos vías. Cada nodo tiene dos enlaces: uno apunta al nodo anterior, o apunta al valor NULL si es el primer nodo; y otro que apunta al nodo siguiente, o apunta al valor NULL si es el último nodo.