Curso: ESTRUCTURA DE DATOS
Tarea: Unidad III. Investigar: El uso de pilas, colas y listas VALOR 10%

Fecha de entrega: lunes, 30 de septiembre de 2025, 00:00

Nicolás REYES MACARIO241u0356@alumno.itssat.edu.mx

Calificado

Rúbrica

Calidad de la Investigación (Contenido)	La investigación es superficial, con fuentes limitadas o irrelevantes, y falta de análisis en muchos aspectos. ***0 puntos***	La investigación es superficial, con fuentes limitadas o irrelevantes, y falta de análisis en muchos aspectos. ***1 puntos***	La investigación es superficial, con fuentes limitadas o irrelevantes, y falta de análisis en muchos aspectos. ***1.5 puntos***	La investigación está muy bien fundamentada, con una excelente selección de fuentes relevantes y actuales. La información es precisa, profunda y aborda todos los aspectos solicitados. ***2 puntos***
Organización y Estructura	El informe carece de una estructura clara. Hay confusión en las secciones o falta de desarrollo en muchas partes. ***1 puntos***	El informe tiene organización básica, pero falta claridad en algunas secciones o la estructura no es lógica. ***2 puntos***	El informe está bien organizado, pero algunas secciones pueden estar mejor estructuradas o desarrolladas. ***3 puntos***	El informe está perfectamente organizado, con una estructura clara y coherente. Cada sección está bien desarrollada. ***4 puntos***

Claridad y Redacción	El informe tiene varios errores de redacción, ortografía y estructura que dificultan su comprensión. ***0 puntos***	El informe tiene algunos problemas de redacción y organización, dificultando su comprensión. ***1 puntos***	El informe es claro y coherente, pero tiene algunos errores menores de redacción o estilo. ***1.5 puntos***	El informe está redactado de manera clara, coherente, y sin errores ortográficos. El lenguaje es adecuado para el contexto académico. ***2 puntos***
Conclusiones	Las conclusiones y recomendaciones son débiles, irrelevantes o mal justificadas. ***0 puntos***	Las conclusiones son generales, y las recomendaciones son vagas o poco claras. ***1 puntos***	Las conclusiones están bien fundamentadas, pero las recomendaciones podrían ser más detalladas o innovadoras. ***1.5 puntos***	Las conclusiones están bien fundamentadas en los resultados obtenidos, y las recomendaciones son muy relevantes, prácticas y aplicables. ***2 puntos***

Calificación actual en el libro

10.00

INSTITUTO TECNOLÓGICO SUPERIOR DE SAN ANDRÉS TUXTLA

ALUMNO: Nicolas Reyes Macario

MATERIA: Estructura de Datos

DOCENTE: Juan Rafael González Cadena

GRUPO: 310-A

FECHA: 30/09/2025

Índice

1. Introducción

En el desarrollo de software, las estructuras de datos organizan la información para que sea procesada eficientemente.[3] De manera sorprendente, los seres humanos aplicamos estas mismas estructuras de forma intuitiva. Entender cómo funcionan las pilas, colas y listas en la vida diaria nos permite comprender mejor la lógica detrás de la eficiencia operativa y la resolución de problemas.

2. La Pila (Stack): El modelo LIFO

Una pila es una estructura donde el último elemento en entrar es el primero en salir (**LIFO**: *Last-In, First-Out*).[4] En la vida diaria, este modelo se aplica en situaciones de apilamiento físico o procesos de interrupción.

- **Platos en la cocina:** Cuando lavamos y secamos platos, los colocamos uno sobre otro. Al necesitarlos, siempre tomamos el que está en la parte superior (el último que pusimos).
- **Historial de navegación y el botón "Atrás":** En internet, cada página que visitas se "apila" sobre la anterior. Cuando presionas "atrás", estás sacando el elemento superior de la pila para ver el anterior.
- **Deshacer (Ctrl + Z):** En un editor de texto, las acciones se guardan en una pila. La última palabra escrita es la primera que se borra al deshacer.

3. La Cola (Queue): El modelo FIFO[5]

La cola sigue el principio de que el primero en llegar es el primero en ser atendido (**FIFO**: *First-In, First-Out*).[6] Es la base de la justicia social en la distribución de servicios.

- **Trámites bancarios o supermercado:** La fila para pagar es el ejemplo más puro. Quien llegó primero sale primero del sistema.[7]
- **Impresoras de oficina:** Cuando varias personas envían documentos a imprimir, la impresora los gestiona en una cola de impresión. No se imprime el último que llegó, sino el primero en la lista de espera.
- **Lavaderos de autos automáticos:** Los vehículos entran en una secuencia lineal y salen en el mismo orden exacto.

4. La Lista (List): Flexibilidad dinámica

A diferencia de las pilas y colas, las listas son flexibles.[8] Podemos añadir, eliminar o consultar elementos en cualquier posición (inicio, medio o fin).[9]

- **Lista del supermercado:** Es una lista enlazada de elementos.[10] No importa el orden estricto en que los anotaste; puedes tachar "leche" aunque esté en medio de la lista o añadir "pan" al principio si se te olvidó.

- **Playlists de música:** Una lista de reproducción permite saltar canciones, mover el orden de los temas o eliminar una canción específica sin afectar la existencia de las demás.
- **Contactos en el teléfono:** Es una lista ordenada (usualmente alfabética) donde puedes insertar un nuevo contacto en cualquier punto de la "L" a la "Z".

5. Cuadro comparativo de aplicaciones

Estructura	Principio	Ejemplo Físico	Ejemplo Digital
Pila (Stack)	LIFO (Último en entrar, primero en salir)	Pila de ropa doblada.	Botón "Atrás" del navegador.
Cola (Queue)	FIFO (Primero en entrar, primero en salir)	Fila en el cine.	Procesamiento de correos electrónicos.
Lista (List)	Acceso Aleatorio / Flexible	Agenda de contactos.	Gestión de tareas (To-Do list).

6. Conclusión

La eficiencia de nuestra rutina diaria depende, en gran medida, de elegir la estructura adecuada para cada tarea. Usar una "pila" para atender clientes en un banco sería injusto (el último en llegar se iría primero), mientras que usar una "cola" para lavar platos sería físicamente imposible. La informática simplemente ha tomado estos modelos naturales y los ha convertido en algoritmos para optimizar el mundo digital.

7. Fuentes de información

- **Sedgewick, R., & Wayne, K. (2025).** *Algorithms and Data Structures: A Practical Approach*. Addison-Wesley.
- **Cormen, T. H. (2024).** *Introduction to Algorithms (4th Edition)*. MIT Press.
- **Real Python.** *Common Data Structures in Python*. [En línea]
- **Khan Academy.** *Linear Data Structures in Everyday Life*.

LISTA DE COTEJO PARA PROGRAMA DE CÓMPUTO

INSTITUTO TECNOLÓGICO SUPERIOR DE SAN ANDRÉS TUXTLA	NOMBRE DEL CURSO: **ESTRUCTURA DE DATOS**
NOMBRE DEL DOCENTE: JUAN RAFAEL GONZÁLEZ CADENA	FIRMA DEL DOCENTE

DATOS GENERALES DEL PROCESO DE EVALUACIÓN		
NOMBRE DEL ALUMNO: **NICOLAS REYES MACARIO**		No. DE CONTROL:
PRODUCTO: **UNIDAD 3** Programa de cómputo	FECHA:	PERIODO ESCOLAR: AGO – DIC 2025

INSTRUCCIONES DE APLICACIÓN

Revisar las actividades que se solicitan y marque con una X en los apartados "SI" cuando la evidencia se cumple; en caso contrario marque "NO". En la columna "OBSERVACIONES" escriba indicaciones que puedan ayudar al alumno a saber cuáles son las condiciones no cumplidas, si fuese necesario.

VALOR DEL REACTIVO	CARACTERÍSTICA A CUMPLIR (REACTIVO)	CUMPLE		OBSERVACIONES
		SI	NO	
3%	Presentación El trabajo cumple con los requisitos de: a. Buena presentación	X		
2%	b. No tiene faltas de ortografía	X		
2%	e. Maneja el lenguaje técnico apropiado	X		
15%	**Desarrollo**: Sigue una metodología y sustenta todos los pasos que se realizaron al aplicar los conocimientos obtenidos, es analítico y bien ordenado.	X		
15%	**Resultados**: Cumplió totalmente con el objetivo esperado, tiene aplicaciones concretas	X		
3%	**Responsabilidad**: Entregó el reporte en la fecha y hora señalada.	X		
40%	CALIFICACIÓN	**40%**		

Prácticas de Estructura de Datos.

ALUMNO: Nicolas Reyes Macario

Programa 1.

Realizar un programa en C++ utilizando LISTAS, que simule un estacionamiento, el programa debe de tener las siguientes opciones: 1 Ocupar, 2 Salir, 3. Consultar y 4 Salir.

PROGRAMA

```
#include <iostream>
#include <list>
using namespace std;

int main() {
   bool est[10] = {false};
   int menu = 0;
   int lugar = 0;

   while (menu != 4){
      cout << "ESTACIONAMIENTO" << endl;
      cout << "Selecciona opcion... " << endl;
      cout << "1. Ocupar lugar" << endl;
      cout << "2. Consulta" << endl;
      cout << "3. Sacar" << endl;
      cout << "4. Salir" << endl;
      cin >> menu;

      switch(menu){
         case 1:
            cout << "Que lugar deseas ocupar?" << endl;
            cin >> lugar;
            if(est[lugar - 1] == true){
               cout << "Lugar Ocupado" << endl;
            }
            else{
               est[lugar - 1] = true;
               cout << "Lugar " << lugar << " ahora esta ocupado" << endl;
            }
            break;

         case 2:
            for(int i = 0; i < 10; i++){
               if(est[i] == true){
                  cout << "Lugar: " << i + 1 << " Ocupado" << endl;
               }
```

```
            else{
               cout << "Lugar: " << i + 1 << " Desocupado" << endl;
            }
         }
         break;

      case 3:
         cout << "Que lugar deseas desocupar?" << endl;
         cin >> lugar;
         if(est[lugar - 1] == false){
            cout << "Ese lugar ya estaba libre" << endl;
         }
         else{
            est[lugar - 1] = false;
            cout << "Lugar " << lugar << " ahora esta libre" << endl;
         }
         break;

      case 4:
         cout << "Saliendo..." << endl;
         break;
   }
  }
}
```

Programa 2

Realizar un programa en C** que realice las funciones básicas de una COLA

```
#include <iostream>
using namespace std;

int main( ){
  int cola [5];
  int t = 0, b = 0, op = 0;

  while (op !=4)
  {
  cout << "1.- Insertar datos: " <<endl;
  cout << "2.- Elimar datos: " <<endl;
  cout << "3.- Recorer cola. " <<endl;
  cout << "4.- Salir. " <<endl;
  cin >> op;

  switch (op)
    {
    case 1:
```

```
            if (t < 5){
            cout << "Ingrese un elemento: ";
            cin >> cola[t];
            t++;
            }
            else{
            cout << "Cola llena."<<endl;
            }
            break;
        case 2:
            if (t > 0){
                cout << "Elemento eliminado: " << cola[b] << endl;
                cola [b] = 0;
                b++;
            }
            else{
                cout << "No hay datos" << endl;
            }
            break;
        case 3:
            if(t > 0){
                for(int i = 0; i<5; i++)
                {
                    cout << cola[i] << endl;
                }
            }
            else{
                cout << "Cola Vacia" << endl;
            }
            break;
        }
    }
}
```

LISTA DE COTEJO PARA PROGRAMA DE CÓMPUTO

INSTITUTO TECNOLÓGICO SUPERIOR DE SAN ANDRÉS TUXTLA	NOMBRE DEL CURSO: **ESTRUCTURA DE DATOS**
NOMBRE DEL DOCENTE: JUAN RAFAEL GONZÁLEZ CADENA	FIRMA DEL DOCENTE

DATOS GENERALES DEL PROCESO DE EVALUACIÓN

NOMBRE DEL ALUMNO: **NICOLAS REYES MACARIO**		No. DE CONTROL:
PRODUCTO: **EXAMEN DE LA UNIDAD 3** Programa de cómputo	FECHA:	PERIODO ESCOLAR: AGO – DIC 2025

INSTRUCCIONES DE APLICACIÓN

Revisar las actividades que se solicitan y marque con una X en los apartados "SI" cuando la evidencia se cumple; en caso contrario marque "NO". En la columna "OBSERVACIONES" escriba indicaciones que puedan ayudar al alumno a saber cuáles son las condiciones no cumplidas, si fuese necesario.

VALOR DEL REACTIVO	CARACTERÍSTICA A CUMPLIR (REACTIVO)	CUMPLE		OBSERVACIONES
		SI	NO	
3%	Presentación El trabajo cumple con los requisitos de: a. Buena presentación	X		
2%	b. No tiene faltas de ortografía	X		
2%	e. Maneja el lenguaje técnico apropiado	X		
20%	**Desarrollo**: Sigue una metodología y sustenta todos los pasos que se realizaron al aplicar los conocimientos obtenidos, es analítico y bien ordenado.	X		
20%	**Resultados**: Cumplió totalmente con el objetivo esperado, tiene aplicaciones concretas	X		
3%	**Responsabilidad**: Entregó el reporte en la fecha y hora señalada.	X		
40%	CALIFICACIÓN	**50%**		

EXAMEN DE LA UNIDAD III VALOR 50%
ALUMNO: NICOLAS REYES MACARIO

Problema. Realizar un programa que permita guardar en un arreglo el nombre y la edad de personas, debe de usar la instrucción “struct”

```
#include <iostream>
#include <string>
using namespace std;

struct Datos{
   string nombre;
   int edad;
};

int main() {
   int opcion;
   int indi = 0;
   string nombre;
   const int TAM = 100;
   Datos arreglo[TAM];

   do {
      cout << "Que accion desea realizar?\n";
      cout << "1.- Agregar Persona\n";
      cout << "2.- Eliminar Persona\n";
      cout << "3.- Buscar Persona\n";
      cout << "4.- Mostrar base de datos\n";
      cout << "5.- Salir\n";
      cout << "Opcion: ";
      cin >> opcion;
      cin.ignore();

      switch (opcion) {
         case 1:
            cout << "Escribe el nombre = ";
            getline(cin, arreglo[indi].nombre);
            cout << "Escribe tu edad = ";
            cin >> arreglo[indi].edad;
            cin.ignore();
            indi++;
            cout << "Usuario agregado correctamente.\n";
            break;

         case 2:
            if (indi == 0) {
```

```
            cout << "Base de datos vacia...\n";
        } else {
            cout << "Usuarios registrados:\n";
            for (int i = 0; i < indi; i++) {
                cout << i << " : " << arreglo[i].nombre << " : " << arreglo[i].edad << endl;
            }
            cout << "Ingrese el nombre del usuario que desea eliminar: ";
            getline(cin, nombre);
            bool eliminado = false;
            for (int i = 0; i < indi; i++) {
                if (nombre == arreglo[i].nombre) {
                    for (int j = i; j < indi - 1; j++) {
                        arreglo[j] = arreglo[j + 1];
                    }
                    indi--;
                    eliminado = true;
                    cout << "Usuario eliminado correctamente.\n";
                    break;
                }
            }
            if (!eliminado) cout << "Usuario no encontrado (Escribe bien su nombre o no
existe).\n";
        }
        break;

    case 3:
        if (indi == 0) {
            cout << "Base de datos vacia...\n";
        } else {
            cout << "Que usuario desea buscar?: ";
            getline(cin, nombre);
            bool encontrado = false;
            for (int i = 0; i < indi; i++) {
                if (nombre == arreglo[i].nombre) {
                    cout << "Usuario encontrado: " << arreglo[i].nombre
                        << " - Edad: " << arreglo[i].edad << endl;
                    encontrado = true;
                    break;
                }
            }
            if (!encontrado) cout << "Usuario no encontrado.\n";
        }
        break;

    case 4:
        if (indi == 0) {
            cout << "Base de datos vacia...\n";
        } else {
```

```
            cout << "Usuarios registrados:\n";
            for (int i = 0; i < indi; i++) {
               cout << i << " : " << arreglo[i].nombre << " : " << arreglo[i].edad << endl;
            }
         }
         break;

      case 5:
         cout << "Tarea finalizada...\n";
         break;

      default:
         cout << "Opcion invalida, intenta nuevamente.\n";
         break;}
   } while (opcion != 5);

   return 0;
}
```