PILAS Y COLAS

UNIVERSIDAD TECNOLÓGICA DE CIUDAD JUÁREZ

Equipo A:
Ángel Alexis Gutiérrez
Cesar Martin Larrea Torres
Dorian García Estrada
Martin Alejandro Leyva Ortega

Introducción

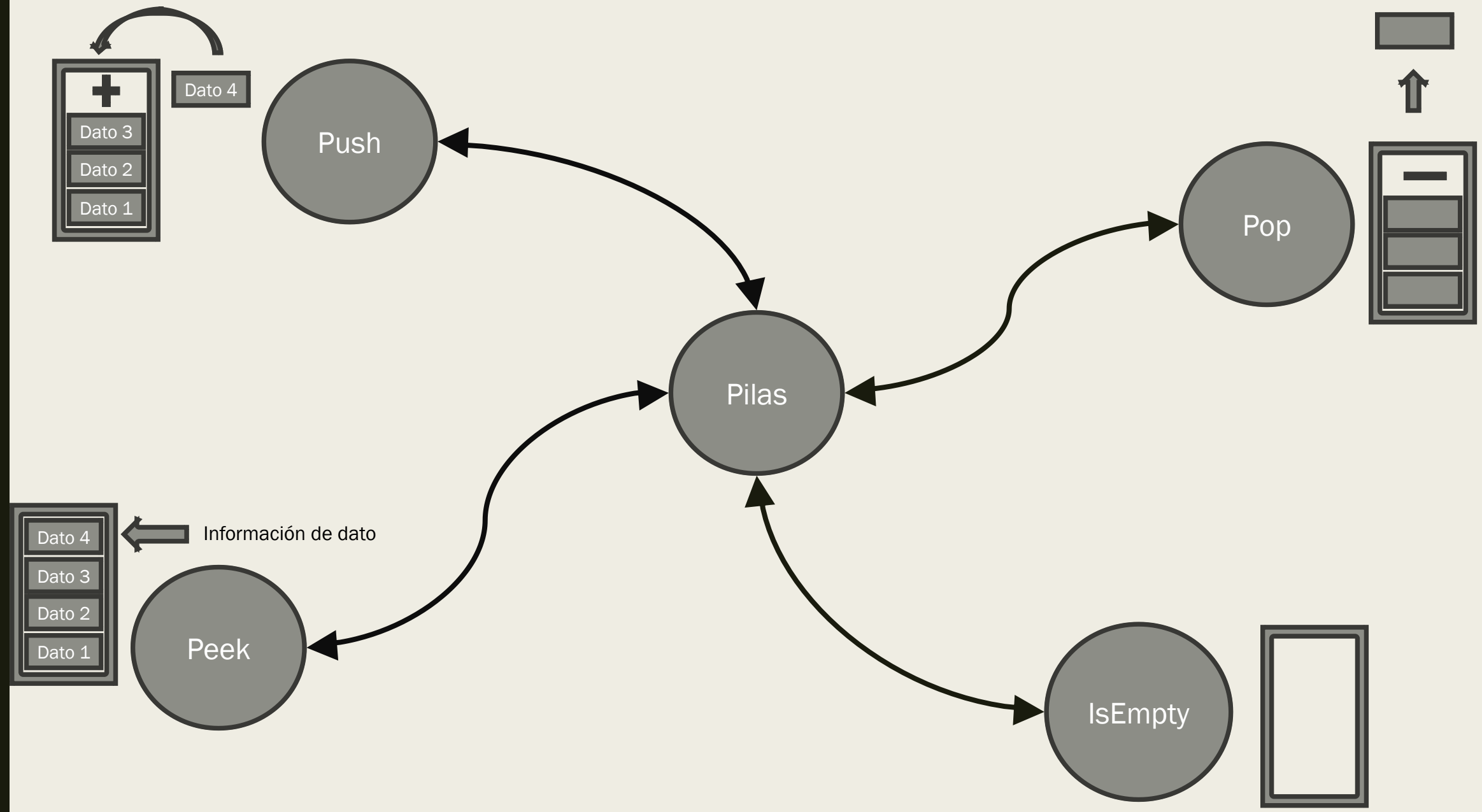
*Pilas:

Una pila es la inserción y eliminación de elementos se realizan solo por un extremo que se denomina cima o tope (top). es un caso particular de la lista donde la inserción y eliminación de elementos se realizan solo por un extremo que se denomina cima o tope (top). Este comportamiento está basado en el principio "último en entrar, primero en salir", también conocido como LIFO, por su nombre en inglés "last-in, first-out", que dicta que el primer elemento que fue añadido a la pila será el último en ser removido de la misma.

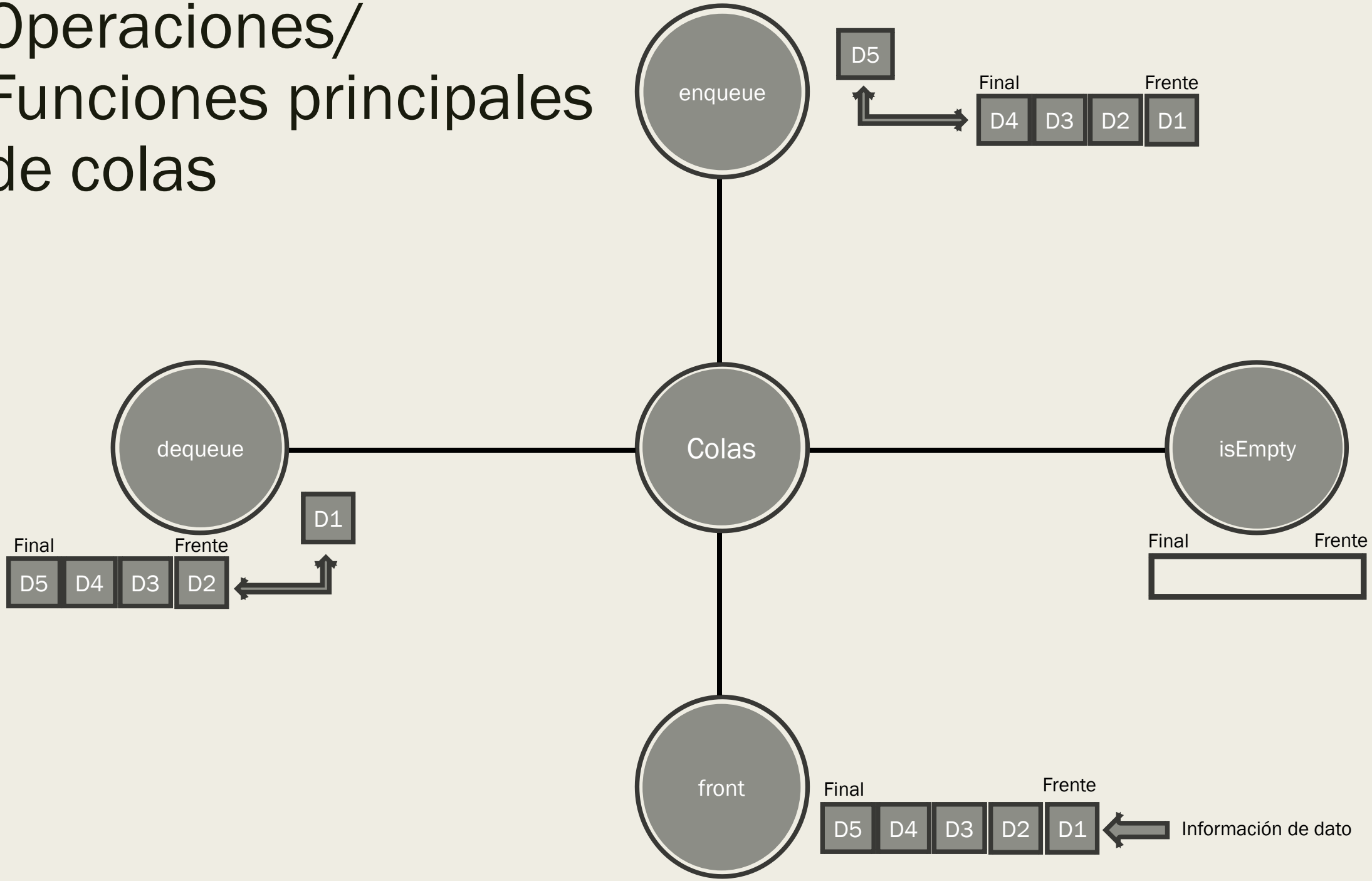
*Colas:

Una pila es la inserción y eliminación de elementos se realizan solo por un extremo que se denomina cima o tope (top). es un caso particular de la lista donde la inserción y eliminación de elementos se realizan solo por un extremo que se denomina cima o tope (top). Este comportamiento está basado en el principio "último en entrar, primero en salir", también conocido como LIFO, por su nombre en inglés "last-in, first-out", que dicta que el primer elemento que fue añadido a la pila será el último en ser removido de la misma.

Operaciones/Funciones principales de pilas



Operaciones/ Funciones principales de colas



Código de funciones principales (Pilas) en c++

```cpp
#include <iostream>
#include <stack>
#include <string>

void mostrar_estado(const std::stack<std::string>& pila) {
    //std::cout << "Estado actual de la pila: ";
    if (pila.empty()) {
        std::cout << "La pila esta vacia." << std::endl;
        return;
    }
    std::stack<std::string> temp = pila; // Usamos una copia para no modificar la original
    while (!temp.empty()) {
        std::cout << temp.top() << " | ";
        temp.pop();
    }
    std::cout << std::endl;
}

int main() {
    std::stack<std::string> pilaAcciones; // Declaramos una pila para guardar strings
    std::string datos[] = {"Carlos", "Ana", "Diego", "Elena", "Bruno", "Fabián", "Laura", "Miguel", "Sofía"};

    std::cout << '\n' <<"--- Demostracion de Pila (LIFO) ---" << std::endl;

    std::cout << "Lista de elementos a insertar en la pila" << std::endl << '\n';
    std::cout << "Carlos, Ana, Diego, Elena, Bruno, Fabián, Laura, Miguel, Sofía" << std::endl << '\n';
```

```cpp
// 1. Simulación de acciones (push)
std::cout << "\n1. Agregando elementos a la pila (push):" << std::endl << '\n';
std::cout << "Agregando elementos...\n" << std::endl;
for (const auto& nombre : datos) {
    pilaAcciones.push(nombre);
    mostrar_estado(pilaAcciones);
}
std::cout << '\n';
std::cout << "- Pila despues de agregar todos los elementos:" << std::endl << '\n';
mostrar_estado(pilaAcciones);
std::cout << '\n' <<"El ultimo en entrar fue: " << pilaAcciones.top() << std::endl << '\n';

// 2. Revisar el elemento superior (peek/top)
std::cout << "\n2. Revisando el elemento en la cima (top):" << std::endl;
if (!pilaAcciones.empty()) { // Verifica si la pila no esta vacia
    std::cout << "El elemento en la cima de la pila es: " << pilaAcciones.top() << std::endl;
}

// 3. Eliminando elementos (pop)
std::cout << "\n3. Eliminando elementos de la pila (pop): \n" << std::endl;
while (!pilaAcciones.empty()) { // Elimina elementos hasta que la pila este vacia
    std::string elementoBorrado = pilaAcciones.top(); // Obtiene el elemento superior
    pilaAcciones.pop(); // Lo elimina de la pila
    mostrar_estado(pilaAcciones);
}

// 4. Verificar si esta vacia (isEmpty/empty)
std::cout << "\n4. Verificando si la pila esta vacia (empty):\n" << std::endl;
if (pilaAcciones.empty()) { // Retorna 'true' si la pila no tiene elementos
    std::cout << "La pila esta vacia. Se han eliminado todos los elementos\n" << std::endl;
}
```

Código de funciones principales (Pilas) en Python

```python
def mostrar_estado(pila):
    print("Estado actual de la pila:", " | ".join(pila) if pila else "La pila esta vacia.")

# 1. Simulación de acciones (push)
print("--- Demostracion de Pila (LIFO) ---")
pila_acciones = []  # Creamos una lista vacia que funcionara como pila
datos = ["Carlos", "Ana", "Diego", "Elena", "Bruno", "Fabián", "Laura", "Miguel", "Sofía"]

print("\n1. Agregando elementos a la pila (push):")
for nombre in datos:
    print(f"Agregando '{nombre}'...")
    pila_acciones.append(nombre)  # Agrega un elemento al final de la lista (cima de la pila)
    mostrar_estado(pila_acciones)

print("\n2. Revisando el elemento superior (peek/top):")
if pila_acciones:
    print(f"El elemento en la cima de la pila es: {pila_acciones[-1]}") # El ultimo elemento de la lista

# 3. Eliminando elementos (pop)
print("\n3. Eliminando elementos de la pila (pop):")
while pila_acciones:
    elemento_borrado = pila_acciones.pop() # Elimina y retorna el ultimo elemento de la lista
    print(f"Eliminando '{elemento_borrado}'...")
    mostrar_estado(pila_acciones)

# 4. Verificar si esta vacia (isEmpty/empty)
print("\n4. Verificando si la pila esta vacia:")
if not pila_acciones:
    print("La pila esta vacia. Se han eliminado todas las acciones.")
```

Código de funciones principales (Colas) en c++

```cpp
#include <iostream>
#include <queue>
#include <string>

void mostrar_estado(const std::queue<std::string>& cola) {
    if (cola.empty()) {
        return;
    }
    std::queue<std::string> temp = cola; // Usamos una copia para no modificar la original
    while (!temp.empty()) {
        std::cout << temp.front() << " | ";
        temp.pop();
    }
    std::cout << std::endl;
}

int main() {
    std::queue<std::string> colaImpresion; // Declaramos una cola para guardar strings
    std::string datos[] = {"Carlos", "Ana", "Diego", "Elena", "Bruno", "Fabián", "Laura", "Miguel", "Sofía"};

    std::cout << "--- Demostracion de Cola (FIFO) ---" << std::endl;

    std::cout << "\nLista de elementos a insertar en la cola" << std::endl << '\n';
    std::cout << "Carlos, Ana, Diego, Elena, Bruno, Fabián, Laura, Miguel, Sofía" << std::endl << '\n';
```

```cpp
// 1. Encolar elementos (enqueue/push)
std::cout << "\n1. Agregando elementos a la cola (push):" << std::endl;
std::cout << "Agregando elementos...\n" << std::endl;
for (const auto& nombre : datos) {
    //std::cout << "Agregando '" << nombre << "' a la cola..." << std::endl;
    colaImpresion.push(nombre); // Agrega un elemento al final de la cola
    mostrar_estado(colaImpresion);
}
std::cout << "\nCola despues de agregar todos los elementos:\n" << std::endl;
mostrar_estado(colaImpresion);
std::cout << "\nEl primero en la cola es: " << colaImpresion.front() << std::endl;

// 2. Revisar el elemento frontal (front)
std::cout << "\n2. Revisando el elemento al frente (front):" << std::endl;
if (!colaImpresion.empty()) {
    std::cout << "El elemento al frente de la cola es: " << colaImpresion.front() << std::endl;
}

// 3. Desencolar elementos (dequeue/pop)
std::cout << "\n3. Eliminando elementos de la cola (pop):\n" << std::endl;
while (!colaImpresion.empty()) { // Elimina elementos hasta que la cola este vacia
    std::string elementoBorrado = colaImpresion.front(); // Obtiene el elemento del frente
    colaImpresion.pop(); // Lo elimina de la cola
    mostrar_estado(colaImpresion);
}

// 4. Verificar si esta vacia (isEmpty/empty)
std::cout << "\n4. Verificando si la cola esta vacia (empty):\n" << std::endl;
if (colaImpresion.empty()) {
    std::cout << "La cola esta vacia. Todos los documentos han sido impresos\n" << std::endl;
}
```

Código de funciones principales (Colas) en Python

```python
from collections import deque

def mostrar_estado(cola):
    print("Estado actual de la cola:", " | ".join(list(cola)) if cola else "La cola esta vacia.")

# 1. Encolar elementos (enqueue/push)
print("--- Demostracion de Cola (FIFO) ---")
cola_impresion = deque()  # Creamos una cola usando deque
datos = ["Carlos", "Ana", "Diego", "Elena", "Bruno", "Fabián", "Laura", "Miguel", "Sofía"]

print("\n1. Agregando elementos a la cola (enqueue):")
for nombre in datos:
    print(f"Agregando '{nombre}' a la cola...")
    cola_impresion.append(nombre)  # Agrega un elemento al final de la cola
    mostrar_estado(cola_impresion)

print("\n2. Revisando el elemento frontal (front):")
if cola_impresion:
    print(f"El elemento al frente de la cola es: {cola_impresion[0]}") # El primer elemento de la cola

# 3. Desencolar elementos (dequeue)
print("\n3. Eliminando elementos de la cola (dequeue):")
while cola_impresion:
    elemento_borrado = cola_impresion.popleft() # Elimina y retorna el elemento del frente de la cola
    print(f"Imprimiendo y eliminando '{elemento_borrado}'...")
    mostrar_estado(cola_impresion)

# 4. Verificar si esta vacia (isEmpty)
print("\n4. Verificando si la cola esta vacia:")
if not cola_impresion:
    print("La cola esta vacia. Todos los documentos han sido impresos.")
```

Prueba de concepto (PoC) en c++

Funcionamiento sin librería(Manual):

```
**************************************************
      PRUEBA DE CONCEPTO: COLA DE IMPRESION
**************************************************
1. Probar con dataset normal (desde dataset.txt)
2. Probar rendimiento con 1000 datos (desde dataset1000.txt)
3. Probar rendimiento con 20000 datos (desde datasetX.txt)
4. Salir
Elige una opcion: 1

Cargando datos desde 'dataset.txt'...

--- 1. Probando la Cola Manual ---
Agregando trabajos a la cola:
  -> [Inicio] Carlos - Ana - Diego - Elena - Bruno - Fabián - Laura - Miguel - sofía - [Final]

Procesando todos los trabajos de la cola manual...
Atendiendo a: Carlos
  -> [Inicio] Ana - Diego - Elena - Bruno - Fabián - Laura - Miguel - sofía - [Final]
Atendiendo a: Ana
  -> [Inicio] Diego - Elena - Bruno - Fabián - Laura - Miguel - sofía - [Final]
Atendiendo a: Diego
  -> [Inicio] Elena - Bruno - Fabián - Laura - Miguel - sofía - [Final]
Atendiendo a: Elena
  -> [Inicio] Bruno - Fabián - Laura - Miguel - sofía - [Final]
Atendiendo a: Bruno
  -> [Inicio] Fabián - Laura - Miguel - sofía - [Final]
Atendiendo a: Fabián
  -> [Inicio] Laura - Miguel - sofía - [Final]
Atendiendo a: Laura
  -> [Inicio] Miguel - sofía - [Final]
Atendiendo a: Miguel
  -> [Inicio] sofía - [Final]
Atendiendo a: sofía
  -> [COLA VACIA]
```

Funcionamiento con librería:

```
--- 2. Probando la Cola con Libreria (std::queue) ---
Agregando trabajos a la cola (con push):
  -> Se agregaron 9 trabajos.

Procesando todos los trabajos de la cola de libreria...
Atendiendo a: Carlos
  -> Quedan 8 trabajos.
Atendiendo a: Ana
  -> Quedan 7 trabajos.
Atendiendo a: Diego
  -> Quedan 6 trabajos.
Atendiendo a: Elena
  -> Quedan 5 trabajos.
Atendiendo a: Bruno
  -> Quedan 4 trabajos.
Atendiendo a: Fabián
  -> Quedan 3 trabajos.
Atendiendo a: Laura
  -> Quedan 2 trabajos.
Atendiendo a: Miguel
  -> Quedan 1 trabajos.
Atendiendo a: sofía
  -> Quedan 0 trabajos.

Presiona Enter para continuar...
```

PoC: utilización de colas

Prueba de concepto (PoC) en c++

PoC: utilización de pilas

```
**************************************************
      PRUEBA DE CONCEPTO: PILA DESHACER/REHACER
**************************************************
1. Demostracion funcional (dataset.txt)
2. Comparacion de rendimiento (dataset1000.txt)
3. Salir
Elige una opcion: 1

Cargando datos desde 'dataset.txt'...

--- 1. Probando la Pila Manual ---
a) Realizando acciones (agregando nombres):
Pila Deshacer:  -> [Cima] sofía | Miguel | Laura | Fabián | Bruno | Elena | Diego | Ana | Carlos | [Fondo]
Pila Rehacer:  -> [PILA VACIA]

b) Deshaciendo la ultima accion:
Pila Deshacer:  -> [Cima] Miguel | Laura | Fabián | Bruno | Elena | Diego | Ana | Carlos | [Fondo]
Pila Rehacer:  -> [Cima] sofía | [Fondo]

c) Rehaciendo la accion:
Pila Deshacer:  -> [Cima] sofía | Miguel | Laura | Fabián | Bruno | Elena | Diego | Ana | Carlos | [Fondo]
Pila Rehacer:  -> [PILA VACIA]


--- 2. Probando la Pila de Libreria (std::stack) ---
a) Realizando acciones (con push):
  -> 9 acciones en Deshacer, 0 en Rehacer.

b) Deshaciendo la ultima accion (con top y pop):
  -> 8 acciones en Deshacer, 1 en Rehacer.

c) Rehaciendo la accion:
  -> 9 acciones en Deshacer, 0 en Rehacer.
```

Punto:
1.-Funcionalidad sin librería (Manual):

Punto:
2.-Funcionalidad con librería:

Prueba de concepto (PoC) en python

Funcionamiento sin librería(Manual):

```
**************************************************
      PRUEBA DE CONCEPTO: COLA DE IMPRESION (PYTHON)
**************************************************
1. Probar con dataset normal (desde dataset.txt)
2. Probar rendimiento con 1000 datos (desde dataset1000.txt)
3. Probar rendimiento con 20000 datos (desde dataset20000.txt)
4. Salir
Elige una opcion: 1

Cargando datos desde 'dataset.txt'...

--- 1. Probando la Cola Manual ---
Agregando trabajos a la cola:
  -> [Inicio] Carlos - Ana - Diego - Elena - Bruno - Fabián - Laura - Miguel - sofía - [Final]

Procesando todos los trabajos de la cola manual...
Atendiendo a: Carlos
  -> [Inicio] Ana - Diego - Elena - Bruno - Fabián - Laura - Miguel - sofía - [Final]
Atendiendo a: Ana
  -> [Inicio] Diego - Elena - Bruno - Fabián - Laura - Miguel - sofía - [Final]
Atendiendo a: Diego
  -> [Inicio] Elena - Bruno - Fabián - Laura - Miguel - sofía - [Final]
Atendiendo a: Elena
  -> [Inicio] Bruno - Fabián - Laura - Miguel - sofía - [Final]
Atendiendo a: Bruno
  -> [Inicio] Fabián - Laura - Miguel - sofía - [Final]
Atendiendo a: Fabián
  -> [Inicio] Laura - Miguel - sofía - [Final]
Atendiendo a: Laura
  -> [Inicio] Miguel - sofía - [Final]
Atendiendo a: Miguel
  -> [Inicio] sofía - [Final]
Atendiendo a: sofía
  -> [COLA VACIA]
```

Funcionamiento con librería:

```
--- 2. Probando la Cola con Libreria (collections.deque) ---
Agregando trabajos a la cola (con append):
  -> Se agregaron 9 trabajos.

Procesando todos los trabajos de la cola de libreria...
Atendiendo a: Carlos
  -> Quedan 8 trabajos.
Atendiendo a: Ana
  -> Quedan 7 trabajos.
Atendiendo a: Diego
  -> Quedan 6 trabajos.
Atendiendo a: Elena
  -> Quedan 5 trabajos.
Atendiendo a: Bruno
  -> Quedan 4 trabajos.
Atendiendo a: Fabián
  -> Quedan 3 trabajos.
Atendiendo a: Laura
  -> Quedan 2 trabajos.
Atendiendo a: Miguel
  -> Quedan 1 trabajos.
Atendiendo a: sofía
  -> Quedan 0 trabajos.

Presiona Enter para continuar...
```

PoC: utilización de colas

Prueba de concepto (PoC) en python

Funcionamiento sin librería(Manual):

```
**************************************************
      PRUEBA DE CONCEPTO: PILA DESHACER/REHACER (PYTHON)
**************************************************
1. Demostracion funcional (dataset.txt)
2. Comparacion de rendimiento (dataset1000.txt)
3. Salir
Elige una opcion: 1

Cargando datos desde 'dataset.txt'...

--- 1. Probando la Pila Manual ---
a) Realizando acciones (agregando nombres):
Pila Deshacer:
  -> [Cima] sofía | Miguel | Laura | Fabián | Bruno | Elena | Diego | Ana | Carlos | [Fondo]
Pila Rehacer:
  -> [PILA VACIA]

b) Deshaciendo la ultima accion:
Pila Deshacer:
  -> [Cima] Miguel | Laura | Fabián | Bruno | Elena | Diego | Ana | Carlos | [Fondo]
Pila Rehacer:
  -> [Cima] sofía | [Fondo]

c) Rehaciendo la accion:
Pila Deshacer:
  -> [Cima] sofía | Miguel | Laura | Fabián | Bruno | Elena | Diego | Ana | Carlos | [Fondo]
Pila Rehacer:
  -> [PILA VACIA]
```

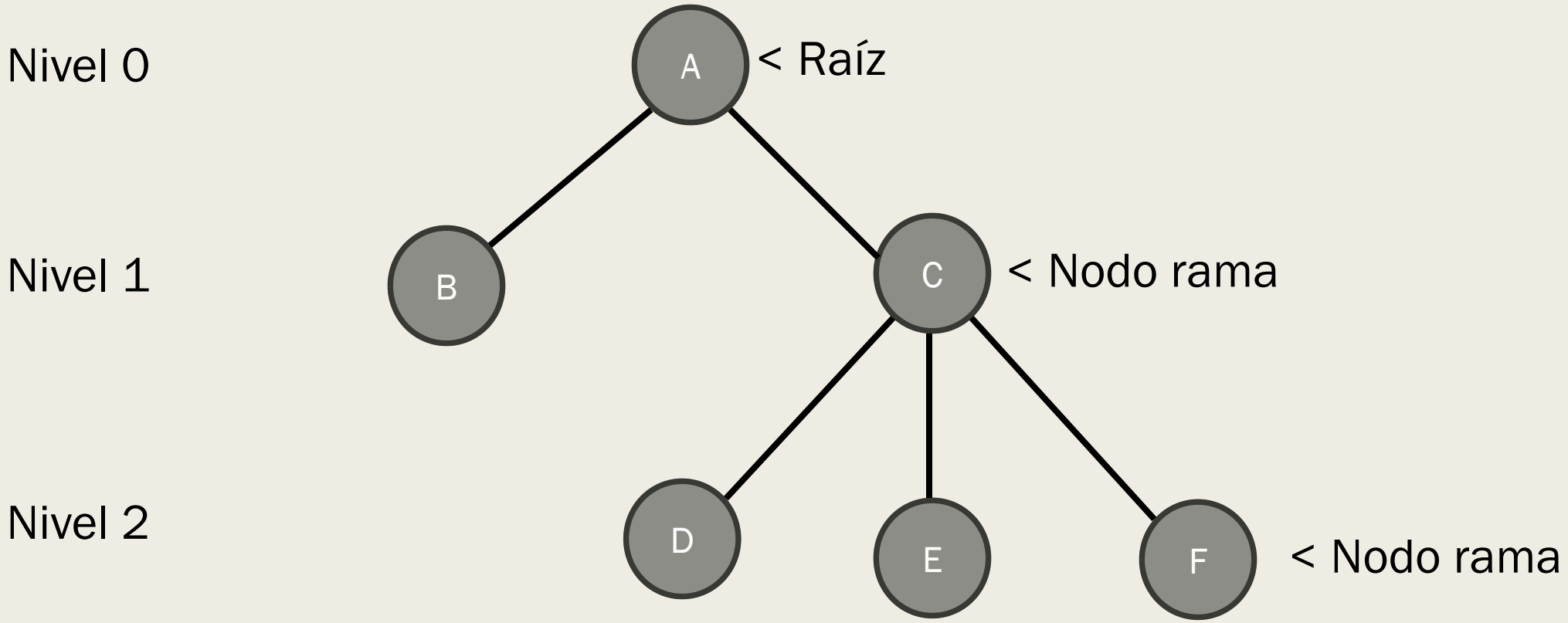
Funcionamiento con librería:

```
--- 2. Probando la Pila de Libreria (listas) ---
a) Realizando acciones (con .append):
  -> 9 acciones en Deshacer, 0 en Rehacer.

b) Deshaciendo la ultima accion (con .pop):
  -> 8 acciones en Deshacer, 1 en Rehacer.

c) Rehaciendo la accion:
  -> 9 acciones en Deshacer, 0 en Rehacer.
```

PoC: utilización de pilas

Arboles Binarios

- Un árbol binario es una estructura de datos en la cual cada nodo puede tener un hijo izquierdo y un hijo derecho. No pueden tener más de dos hijos (de ahí el nombre "binario"). Si algún hijo tiene como referencia a null, es decir que no almacena ningún dato, entonces este es llamado un nodo externo.



Conclusión

En resumen, a lo largo de esta presentación hemos explorado dos de las estructuras de datos fundamentales:
Las Pilas operan bajo el principio LIFO (Last-In, First-Out), el último en entrar es el primero en salir, como una pila de platos. Por otro lado, las Colas siguen el principio FIFO (First-In, First-Out), donde el primero en llegar es el primero en ser atendido, similar a una fila de personas.

Sobre los árboles binarios nos han mostrado que las pilas y las colas pueden ser el cimiento para construir estructuras y soluciones más complejas