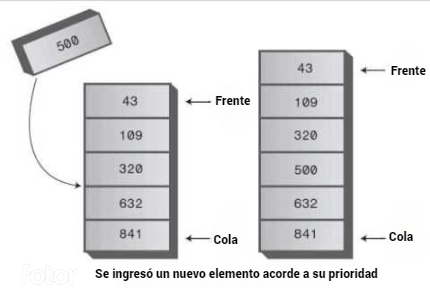
**Aplicaciones: Colas de Prioridad**

1. Definición de Cola de Prioridad

Una cola de prioridad es una estructura de datos abstracta que mantiene una colección de elementos, cada uno asociado con una prioridad. En contraste con una cola convencional, donde los elementos se insertan y se eliminan en el orden en que llegan (FIFO), en una cola de prioridad, los elementos se extraen de acuerdo con su prioridad relativa. Es decir, el elemento con la prioridad más alta será el próximo en ser eliminado.



*Imagen 1: Estructura de la Cola de Prioridad*

**2. Aplicaciones de las Colas de Prioridad:**

Las colas de prioridad tienen una amplia variedad de aplicaciones en el desarrollo de software. Algunos ejemplos destacados incluyen:

1. Algoritmos de Búsqueda de Caminos: En algoritmos como Dijkstra o A\*, las colas de prioridad se utilizan para determinar cuál será el próximo nodo a explorar, priorizando aquellos que tienen la menor distancia acumulada o el menor costo.
2. Planificación de Procesos: En sistemas operativos multitarea, las colas de prioridad se utilizan para asignar tiempo de CPU a los procesos en función de su importancia relativa. Los procesos de alta prioridad tienen acceso preferencial a los recursos del sistema.
3. Gestión de Eventos en Simulaciones: En aplicaciones de simulación, las colas de prioridad se utilizan para programar eventos futuros y garantizar que se procesen en el orden correcto, según su prioridad temporal.
4. Compresión de Datos: Algunos algoritmos de compresión, como el algoritmo Hoffman, utilizan colas de prioridad para construir eficientemente el árbol de codificación que minimiza la longitud total del código.

**2.1 Ejemplo:**

A continuación, presentamos un ejemplo de implementación de una cola de prioridad en C++ utilizando archivos separados.

***Nodo.h:***

#ifndef NODO\_H\_INCLUDED

#define NODO\_H\_INCLUDED

class Nodo {

private:

int dato;

int prioridad;

Nodo\* siguiente;

public:

Nodo(int valor, int prioridad);

int getDato();

int getPrioridad();

Nodo\* getSiguiente();

void setSiguiente(Nodo\* nodo);

};

#endif // NODO\_H\_INCLUDED

***Nodo.cpp:***

#include "Nodo.h"

Nodo::Nodo(int valor, int prioridad) : dato(valor), prioridad(prioridad), siguiente(nullptr) {}

int Nodo::getDato() {

return dato;

}

int Nodo::getPrioridad() {

return prioridad;

}

Nodo\* Nodo::getSiguiente() {

return siguiente;

}

void Nodo::setSiguiente(Nodo\* nodo) {

siguiente = nodo;

}

***ColaPrioridad.h:***

#ifndef COLA\_PRIORIDAD\_H\_INCLUDED

#define COLA\_PRIORIDAD\_H\_INCLUDED

#include "Nodo.h"

class ColaPrioridad {

private:

Nodo\* frente;

Nodo\* fin;

public:

ColaPrioridad();

~ColaPrioridad();

void encolar(int valor, int prioridad);

void desencolar();

void imprimir();

};

#endif // COLA\_PRIORIDAD\_H\_INCLUDED

***ColaPrioridad.cpp:***

#include "ColaPrioridad.h"

#include <iostream>

ColaPrioridad::ColaPrioridad() : frente(nullptr), fin(nullptr) {}

ColaPrioridad::~ColaPrioridad() {

while (frente != nullptr) {

Nodo\* temp = frente;

frente = frente->getSiguiente();

delete temp;

}

}

void ColaPrioridad::encolar(int valor, int prioridad) {

Nodo\* nuevo = new Nodo(valor, prioridad);

if (frente == nullptr || prioridad < frente->getPrioridad()) {

nuevo->setSiguiente(frente);

frente = nuevo;

} else {

Nodo\* actual = frente;

while (actual->getSiguiente() != nullptr && actual->getSiguiente()->getPrioridad() <= prioridad) {

actual = actual->getSiguiente();

}

nuevo->setSiguiente(actual->getSiguiente());

actual->setSiguiente(nuevo);

}

}

void ColaPrioridad::desencolar() {

if (frente != nullptr) {

Nodo\* temp = frente;

frente = frente->getSiguiente();

delete temp;

} else {

std::cout << "La cola esta vacia, no se puede desencolar\n";

}

}

void ColaPrioridad::imprimir() {

Nodo\* temp = frente;

while (temp != nullptr) {

std::cout << temp->getDato() << " ";

temp = temp->getSiguiente();

}

std::cout << "\n";

}

***main.cpp***

#include "ColaPrioridad.cpp"

#include "Nodo.cpp"

#include <iostream>

#include <cstdlib>

int main() {

ColaPrioridad miCola;

int valor, prioridad, opcion;

do {

std::cout << "\n1. Encolar elemento";

std::cout << "\n2. Desencolar elemento";

std::cout << "\n3. Imprimir cola";

std::cout << "\n4. Salir";

std::cout << "\nIngrese la opcion deseada: ";

std::cin >> opcion;

switch(opcion) {

case 1:

system("cls");

std::cout << "Ingrese el valor a encolar: ";

std::cin >> valor;

std::cout << "Ingrese la prioridad: ";

std::cin >> prioridad;

miCola.encolar(valor, prioridad);

break;

case 2:

system("cls");

miCola.desencolar();

break;

case 3:

system("cls");

miCola.imprimir();

break;

}

} while(opcion != 4);

return 0;

}

**3. Conclusiones**

Las colas de prioridad son una herramienta poderosa en la programación y tienen una amplia gama de aplicaciones en diversas áreas, desde algoritmos de búsqueda hasta sistemas operativos y compresión de datos. Comprender cómo funcionan y cómo implementarlas correctamente es esencial para desarrollar software eficiente y escalable.

**4. Referencias**

de la Cueva, V., González, L., & Salinas, E. (2020). *Estructuras de datos y algoritmos fundamentales* (pp. 51–58). Editorial Digital del Tecnológico de Monterrey.

<https://www.google.com.ec/books/edition/Estructuras_de_datos_y_algoritmos_fundam/MXf1DwAAQBAJ?hl=es-419&gbpv=0>