PRACTICA 1

ESTRUCTURAS DE DATOS

Emanuel Baciu y Beatriz Arribas Schudeck

Introducción

El programa siguiente describe la implementación del funcionamiento de una editorial. Imita la gestión de los distintos pedidos solicitados a distintas librerías y el flujo que estos transitan desde su petición hasta su empaquetado y envío.

Lo que se espera, es poder visualizar los comportamientos de las distintas colas y listas dinámicas que conforman la estructura del programa.

Breve descripción de los TAD´s

-Pila

El TAD pila cumple la función de poder almacenar los pedidos que previamente han pasado por la fase de "Listo" a la respectiva caja de la librería de la que inicialmente se hizo el pedido.

-Las operaciones básicas que presenta este TAD son:

- apilar: introduce un pedido en la cima de la pila.
- desapilar: quita el pedido que se encuentra encima de la pila.
- cima: muestra el pedido que esta encima de la pila.
- esVacia: comprueba si la pila contiene algún pedido
- getLength : devuelve el número de pedidos que contiene la pila.

-Implementación con pilas dinámicas

-Cola

El TAD cola resuelve el problema del control de pedidos por fases, es decir, por cada estado de libro existe una cola que los almacena.

-Las operaciones básicas que presenta son:

- encolar: añade un pedido al final de la cola
- desencolar: quita el primero de la cola
- esVacia: comprueba si una cola tiene algún pedido
- getLength: devuelve el número de pedidos de la cola

-Implementación con colas dinámicas

-Pedido

El TAD pedido condensa todos los campos que identifican unívocamente la solicitud de un pedido

-No tiene operaciones propias básicas, pero si funciones que trabajan sobre este TAD.

-Stock

El stock es la estructura que soluciona el almacenamiento de los libros disponibles y facilita la comprobación para modificar el estado de un pedido de "Almacén" a "Imprenta" según se encuentre disponibilidad de ejemplares en el stock.

-Implementación con array estático con limite de valores de "MAX_TITULOS = 10"

-Pedido Stock

El TAD "pedido stock" diferencia únicamente los campos que permiten su búsqueda en el catálogo y la comprobación del número de ejemplares disponible almacenados.

Métodos/Funciones implementadas

Funcionamiento del programa

Resumen funcional

Este programa permite gestionar los pedidos de libros de ciertos establecimientos, permitiendo ver desde su salida del almacén, hasta su paso por imprenta si es que no hay suficientes ejemplares para abastecer el pedido.

Ejecución

Al iniciar el programa, el usuario verá un menú con opciones, de las cuales podrá elegir indistintamente, y se ejecutará la función correspondiente. El ciclo se repite hasta que se pulse la opción de "salir". El usuario tiene que despreocuparse de introducir peticiones por la consola, solo verá una simulación.

Estructura interna

Los pedidos se gestionan a través de una struct "Pedido". Las estructuras que permiten el correcto funcionamiento del flujo de programa son los respectivos structs de "Pila" y "Cola". Para crear el catálogo se utiliza el struct "pedido_stock", una especie de pedido pero solo con ciertos campos selectos como atributos que facilita la implementación de Stock. Métodos como "printQueue" y "printPile" permiten visualizar el estado de los pedidos y en que nodos del proceso se encuentras. El control del menú esta enteramente implementado en "main".

Métodos y funciones

Funciones implementadas

- Cola::esVacia()-> un método que comprueba si la cola está vacía.
- Pila::esVacia()-> un método que comprueba si la pila está vacía.
- Pila::getLength()-> método que devuelve el numero de elementos de una pila.
- Cola::getLength() -> método que devuelve el numero de elementos de una cola.
- genPedido -> genera un pedido pseudoaleatorio.
- genStock()-> permite crear un catálogo de libros.

- findInStock-> comprueba si hay suficientes libros almacenados para satisfacer el numero de ejemplares solicitados.
- mvItemsQueue()->mueve los pedidos a la siguiente cola en caso de las fases intermedias, y a una pila si están listos para ser enviados.
- Los "prints"-> imprimen una cola, pila o el stock, dependiendo del parámetro que se le pase como argumento.

Problemas encontrados y la solución adoptada

Durante la práctica han surgido varios problemas, he aquí los más célebres:

- **Reducción de longitud de pilas**

  Un bucle "for" situado en "editorial.cpp" que mostraba el contenido de la respectiva caja de la librería, reducía progresivamente el contenido de esta y, por lo tanto, no todo el contenido se mostraba.

  Solución: la condición de parada era directamente la devolución de "getLength" por lo que, cada vez que desencolaba para mostrar el pedido, se calculaba de nuevo la longitud de la pila manipulada y la iteración se hacía con varios elementos menos. Solucionado con una variable local pasada como condición de parada de la cual su valor no varía.

- **Atributos privados**

  No nos era posible acceder, por ejemplo, a la cima de la pila o al valor "frente" de la cola.
  Solución: comprobar si la clase Nodo era amiga de la clase Pila y Cola. Tras esta operación, el problema no remitía, por tanto, hicimos la comprobación de frente y cima como método.

- **Problemas en funciones para encontrar variables declaradas en el "main".**

  La función encargada del paso de los pedidos de una cola a otra no encontraba las variables de las respectivas colas declaradas en el main. El error que se mostraba indicaba que se estaban redefiniendo las colas.
  Solución: paso por referencia a las funciones correspondientes

- **Incorrecto funcionamiento de la reducción de stock**

  Tras la comprobación de que sí que existe suficiente stock para abastecer el pedido, la reducción de la diferencia entre el numero de ejemplares y los disponibles no se llevaba a cabo, y el stock se mantenía sin ninguna alteración.
  Solución: el problema detectado era el uso indebido de la variable en la que estábamos almacenando el elemento que se quería manipular. Los valores solo se

alteraban en dicha variable. La solución adoptada fue el simple uso de la posición referida al libro en stock que se quería consultar para poder cambiar directamente su valor en el array.