METODO DE LA INGENIERÍA

Fase 1: Identificación del problema

La cultura de la lectura y el libro ha sido a lo largo de la historia de Cali muy importante, pero ha sufrido un declive en los últimos tiempos, sin embargo ha renacido, es por esta razón que una librería muy importante e innovadora a decidido abrir sus puertas en la sultana del Valle. Esta empresa se ha caracterizado siempre por ser innovadora y eficiente a la hora de atender a los usuarios.

Para este nuevo proyecto tienen planificado toda una serie de elementos que volverán eficientes dicha atención de los clientes desde el momento en que entran a la tienda contando con pantallas táctiles para revisar todos los títulos disponibles hasta el momento en que se efectúa el pago el cual se basa en el orden de tiempo que lleva el cliente en la tienda desde el momento que entro.

- **Definición del problema:**

Se ha requerido el desarrollo de una herramienta que permita a los caleños conocer acerca de cómo sería el funcionamiento de la nueva librería en la ciudad, simulando el proceso de compra desde la salida de la sección 1.

Requerimientos Funcionales

La solución del problema:

R1: Debe simular el proceso de compra de libros desde la salida de la sección 1 pasando por cada una de las secciones, mostrando de esta forma los libros disponibles en la tienda cuando un cliente ingresa, el momento de selección de los clientes y su respectiva recolección de los libros, finalizando con el pago en las cajas que se encuentren disponibles todo esto en el respectivo orden basado en el tiempo que lleva una persona desde que llego a la tienda hasta este punto.

R2: Debe recibir los siguientes datos básicos:

- ISBN del libro.
- Cantidad de ejemplares.
- Estantería donde se encuentra ubicado.

- Cantidad de cajeros a ser utilizados durante la jornada.
- Serie de código o cédulas que representan a los clientes (en el orden que entraron a la tienda).
- Lista de libros por comprador.

R3: Debe informar el orden de salida de los clientes tomando el orden en que llegaron a las cajas y cuantos turnos se van a llevar ahí acorde al número de libros que ha decidido llevar, el valor de cada compra y el orden en que quedaron empacados sus libros.

R4: Debe tener el uso de estructuras de datos cómo lo son pilas, colas y tablas hash para llevar a cabo la solución de todo lo nombrado anteriormente.

Fase 2: Recopilación de la Información

ISBN

¿Qué es?

Un ISBN es un código normalizado internacional para libros (International Standard Book Number). Los ISBN tuvieron 10 dígitos hasta diciembre de 2006 pero, desde enero de 2007, tienen siempre 13 dígitos. Los ISBN se calculan utilizando una fórmula matemática específica e incluyen un dígito de control que valida el código.

Cada ISBN se compone de 5 elementos separados entre sí por un espacio o un guion. Tres de los cinco elementos pueden variar en longitud:

Elemento prefijo – actualmente sólo pueden ser 978 o 979. Siempre tiene 3 dígitos de longitud.

Elemento de grupo de registro –identifica a un determinado país, una región geográfica o un área lingüística que participan en el sistema ISBN. Este elemento puede tener entre 1 y 5 dígitos de longitud.

Elemento del titular – identifica a un determinado editor o a un sello editorial. Puede tener hasta 7 dígitos de longitud.

Elemento de publicación – identifica una determinada edición y formato de un determinado título. Puede ser de hasta 6 dígitos de longitud.

Dígito de control – es siempre el último y único dígito que valida matemáticamente al resto del número. Se calcula utilizando el sistema de Módulo 10 con pesos alternativos de 1 y 3.

¿Para qué sirve un ISBN?

Un ISBN es, en lo esencial, un identificador de producto utilizado por editores, libreros, tiendas online y otros participantes en la cadena comercial para pedidos, listados, registros de venta y control de existencias. El ISBN identifica tanto al titular como a un título específico, su edición y su formato.

¿Qué tipo de publicación identifica un ISBN?

Los ISBN se asignan a las publicaciones monográficas (es decir, de un solo elemento físico y no a las revistas, periódicos u otro tipo de publicaciones seriadas) compuestas de texto.

Cualquier libro que esté disponible para el público, sea para su venta o gratuito, puede identificarse con un ISBN.

Además, las partes individualizadas (como los capítulos) de libros, o los ejemplares o artículos de revistas, diarios o publicaciones seriadas que se distribuyan separadamente pueden también utilizar el ISBN como identificador.

En lo que respecta a los diferentes soportes disponibles, carece de importancia la forma en que el contenido es presentado y distribuido; pero cada formato de producto diferente (como encuadernación rústica, EPUB, .Pdf, etc.) debe ser identificado de manera diferenciada.

Se pueden consultar ejemplos de tipos de productos que llevan ISBN y más información sobre su función y alcance aquí.

Los ISBN, las leyes y los derechos de propiedad intelectual

El ISBN es un identificador y no conlleva ninguna forma de protección legal ni de los derechos de propiedad intelectual. Sin embargo, en algunos países el uso del ISBN para identificar las publicaciones es un requisito legal.

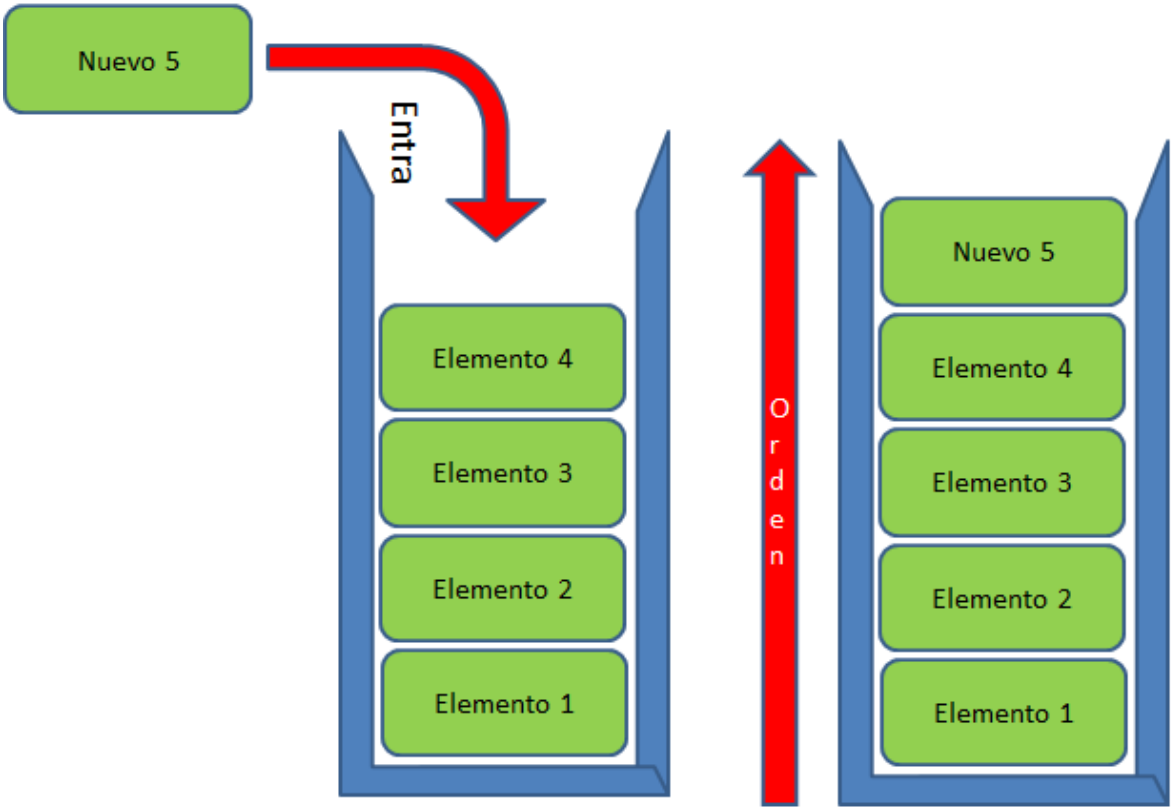
¿Quién debe solicitar los ISBN?

Es el editor del libro el que debe solicitar siempre el ISBN. A efectos del ISBN, el editor es el grupo, organización, compañía o individuo responsable de poner en marcha la producción de una publicación. Normalmente, es también la persona o entidad que sufraga el coste y asume el riesgo económico de hacer el producto disponible. Normalmente no es el impresor, pero puede ser el autor del libro si éste ha decidido publicar el libro por su cuenta.

PILAS

Pila (informática).Una pila (stack en inglés) es una lista ordinal o estructura de datos en la que el modo de acceso a sus elementos es de tipo LIFO (del inglés Last In First Out, último en entrar, primero en salir) que permite almacenar y recuperar datos. Se aplica en multitud de ocasiones en informática debido a su simplicidad y ordenación implícita en la propia estructura. Representación gráfica de una pila

Para el manejo de los datos se cuenta con dos operaciones básicas: apilar (push), que coloca un objeto en la pila, y su operación inversa, retirar (o desapilar, pop), que retira el último elemento apilado.



En cada momento sólo se tiene acceso a la parte superior de la pila, es decir, al último objeto apilado (denominado TOS, Top of Stack en inglés). La operación retirar permite la obtención de este elemento, que es retirado de la pila permitiendo el acceso al siguiente (apilado con anterioridad), que pasa a ser el nuevo TOS.

Por analogía con objetos cotidianos, una operación apilar equivaldría a colocar un plato sobre una pila de platos, y una operación retirar a retirarlo.

Las pilas suelen emplearse en los siguientes contextos:

Evaluación de expresiones en notación postfija (notación polaca inversa).

* Reconocedores sintácticos de lenguajes independientes del contexto

* Implementación de recursividad.

Operaciones

Una pila cuenta con 2 operaciones imprescindibles: apilar y desapilar, a las que en las implementaciones modernas de las pilas se suelen añadir más de uso habitual.

Crear: se crea la pila vacía.

Apilar: se añade un elemento a la pila.(push)

Desapilar: se elimina el elemento frontal de la pila.(pop)

Cima: devuelve el elemento que está en la cima de la pila. (top o peek)

Vacía: devuelve cierto si la pila está vacía o falso en caso contrario.

Implementación

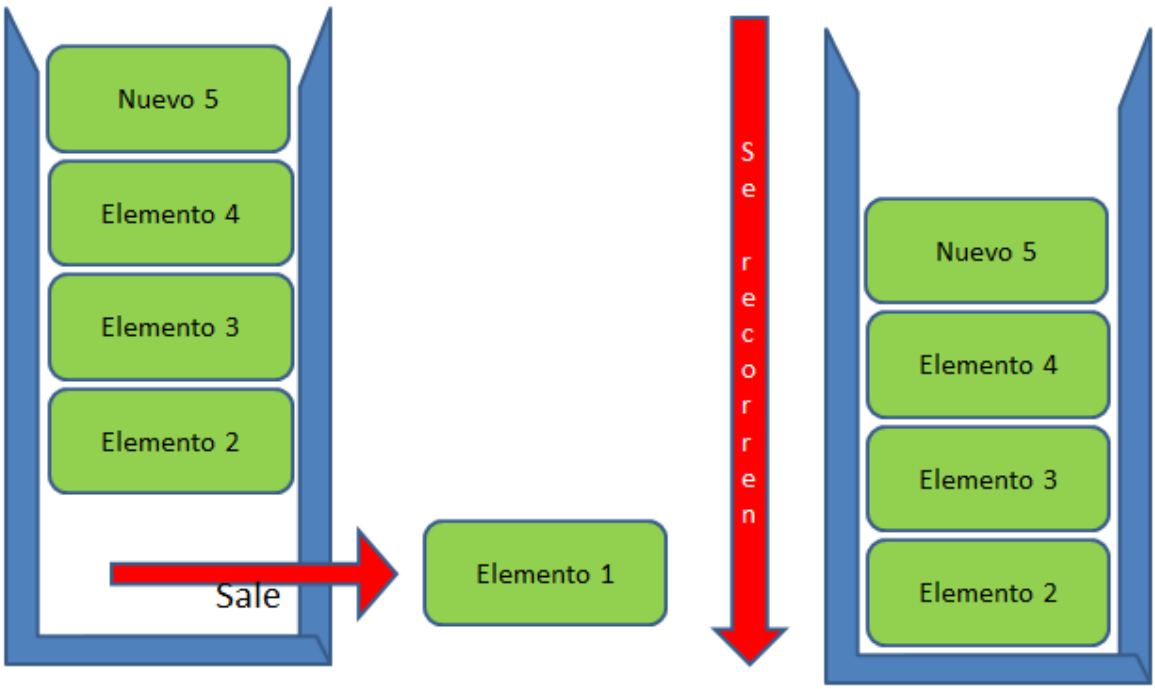
Un requisito típico de almacenamiento de una pila de n elementos es O(n). El requisito típico de tiempo de O(1) las operaciones también son fáciles de satisfacer con un array o con listas enlazadas simples.

La biblioteca de plantillas de C++ estándar proporciona una "pila" clase templated que se limita a sólo apilar/desapilar operaciones. Java contiene una biblioteca de la clase Pila que es una especialización de Vector. Esto podría ser considerado como un defecto, porque el diseño heredado get () de Vector método LIFO ignora la limitación de la Pila.

COLAS

Cola (informática). Una cola es una estructura de datos, caracterizada por ser una secuencia de elementos en la que la operación de inserción push se realiza por un extremo y la operación de extracción pop por el otro. También se le llama estructura FIFO (del inglés First In First Out), debido a que el primer elemento en entrar será también el primero en salir.

Las colas se utilizan en sistemas informáticos, transportes y operaciones de investigación (entre otros), dónde los objetos, personas o eventos son tomados como datos que se almacenan y se guardan mediante colas para su posterior procesamiento. Este tipo de estructura de datos abstracta se implementa en lenguajes orientados a objetos mediante clases, en forma de listas enlazadas.



Operaciones Básicas

Crear: se crea la cola vacía.

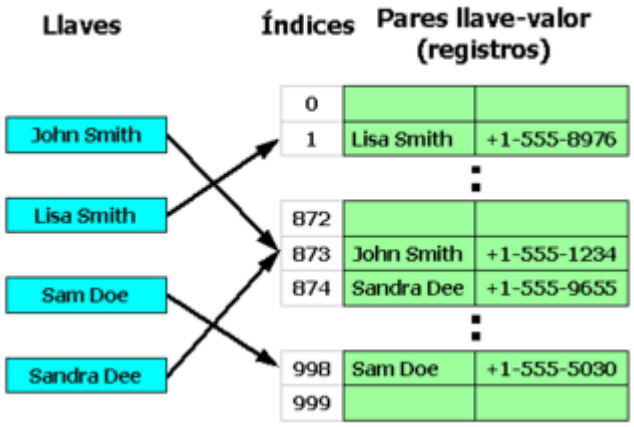
Encolar (añadir, entrar, push): se añade un elemento a la cola. Se añade al final de esta.

Desencolar (sacar, salir, pop): se elimina el elemento frontal de la cola, es decir, el primer elemento que entró.

Frente (consultar, front): se devuelve el elemento frontal de la cola, es decir, el primero elemento que entró.

TABLA HASH

Es una estructura de datos no lineal cuyo propósito final se centra en llevar a cabo las acciones básicas (inserción, eliminación y búsqueda de elementos) en el menor tiempo posible, mejorando las cotas de rendimiento respecto a un gran número de estructuras.



Tablas Hash. Idea general de funcionamiento

Una Tabla Hash es una estructura que puede almacenar una gran cantidad de información y lograr para sus acciones principales tiempos de ejecución Θ(1). Esto supera las cotas comunes de, por ejemplo, las listas enlazadas, cuyos tiempos oscilan aproximadamente por Θ(N).

La idea general de la Tabla Hash es utilizar la capacidad de los arreglos lineales, donde las acciones de inserción y obtención poseen tiempos Θ(1), y asignar ese mismo tiempo a la acción de búsqueda, que en un arreglo sí es de Θ(N).

Fase 3: Búsqueda de soluciones creativas

•La generación de las ideas se desarrollaron de forma conjunta planeando cuales serían las posibles soluciones al problema, donde aceptamos todo tipo de idea, pero con el fin de que su uso fuese lógico en la solución, sin importar si era el más optimo, el más eficiente o algún otro criterio.

1. Mediante el uso de algoritmos de ordenamiento organizar cada uno de los clientes para que vayan entrando a las cajas de pago.
2. Organizar cada uno de los clientes en arreglos para así llevar toda su información y sus registros.
3. Utilizar las estructuras de datos pila y cola para almacenar a cada uno de los clientes acorde a su llegada a la tienda y el momento en donde deben cancelar en caja, utilizando la estructura respectiva en casos determinados.
4. Utilizar tablas hash para guardar cada uno de los libros con sus respectivas claves almacenadas.
5. Buscar algoritmos existentes que nos permitan facilitar la implementación con los conocimientos ya obtenidos en APO2.
6. Mediante el uso de árboles binarios guardar cada uno de los clientes acorde a su tiempo de llegada.
7. Mediante el uso de listas enlazadas organizar los libros con sus respectivos ISBN.

Fase 4: Transición de las Ideas a los Diseños Preliminares

Las siguientes ideas las descartaremos tomando como base de criterio el desarrollo por completo y de forma efectiva lo que se nos está pidiendo en este caso con el desarrollo de la factorización de polinomios para hallar sus raíces, no obstante, en cada una de las ideas descartadas que daremos a continuación argumentaremos por qué se tomó la decisión.

Alternativa 1: Uso de algoritmos de ordenamiento

-El uso de métodos de ordenamiento queda totalmente obsoleto en este caso ya que los clientes al entrar pasan por diferentes tipos de lugares en donde su estado se va transformando, entonces depende de más variables que con algoritmos de ordenamiento no puedo emplear su orden de llegada a las cajas.

Alternativa 5-6: Uso de listas enlazadas

-El uso de Listas es totalmente absurdo y aunque puede llegar a ser interesante no se empleará su uso como fundamento del desarrollo de la solución.

Alternativas 5-7: Uso de árboles binarios

-Por el mismo argumento anterior se descarta la utilización de árboles binarios, ya que este método o vía podrá servir en algo, pero en nada al fin que es organizar los clientes y libros.

Diseños Preliminares

TAD Queue
Queue = { Node<T> first, Node<T> end, int size }
{inv: }
Operaciones Primitivas: -End: → Node<T> -Enqueue: Node<T> → Node<T> -Dequeue: → Node<T> -isEmpty: → Booleano -size: → Entero -first: → Node<T>

End() “Retorna el último elemento de la cola* {pre: } {post: Null si la cola esta vacía, de lo contrario el último elemento de la cola}

Enqueue(T o) “Encola-inserta un elemento-objetos en la cola” {pre: } {post: o = { nuevoNodo: Nodo<T>}}

Dequeue() “Obtiene un elemento que se encuentra en la cola desencolando cumpliendo las propiedades” {pre: } {post: o = {first.getElement} , size = {size--} }

isEmpty()

“Verifica si la cola se encuentra vacía o no”

{pre: }

{post: TRUE si la cola está vacía FALSE si es el caso contrario.

Size()

“Da el tamaño de la cola”

{pre: Cola existente.}

{post: size= { tamaño de la cola(int) } }

First()

“Da el primer elemento de la cola”

{pre: }

{post: first = { first.getElement() } }

TAD Stack
Stack = { Node<T> end, Entero size}
{inv: }
Operaciones Primitivas: -Push: Node<T> → Node<T> -Pop: → Node<T> -Size: → Entero -isEmpty: → Booleano -Peek: → Node<T>

Push(T elem)

“Agrega el elemento elem a la pila”

{pre: pila vacía ya creada}

{post: elem{insertada en la pila} }

Pop() “Saca el elemento elem a la pila” {pre: La pila debe tener elementos} {post: T o={end.getElement()} }

Size() “Devuelve la cantidad de elementos de la pila” {pre: La pila debe existir} {post: size{int} }

isEmpty() “Verifica si la pila se encuentra vacía o no” {pre: } {post: TRUE si la cola está vacía FALSE si es el caso contrario.

Peek() “Devuelve el elemento ubicado al tope de la pila sin sacarlo” {pre: } {post: end={ end.getElement() } }

TAD Shelving
Shelving = { String k, int v}
{inv: }
Operaciones Primitivas: -hashFunction: String → Entero -getBooks: Arreglo de Libros → books -addBook: entero x entero x entero → book add -insert: String x entero → void - delete: String → void

hashFuction(String k)

“Realiza la función hash a partir de una clave”

{pre: clave existente}

{post: position = {Interget.parseInt(k) } }

getBooks()

“obtiene los libros que se encuentran en una ArrayList”

{pre: ArrayList con elementos}

{post: books }

addBook(int k, int valu, int Price)

“agrega un libro con su clave, su valor y el precio”

{pre: }

{post: libro agregado con sus respectivas características}

Insert(String key, Interger value)

“Inserta en la tabla hash”

{pre: tabla hash existente}

{post: valor y posición insertada}

Delete(String key)

“Elimina en la tabla hash con la clave”

{pre: tabla hash existente y clave ya en la tabla }

{post: elemento respecto a la clave ya eliminado}

Fase 5: Evaluación y Selección de la Mejor Solución

Criterios

Estos son los principios por los cuales serán evaluadas las ideas y donde escogeremos las apropiadas para el desarrollo de la solución del problema de forma definitiva.

Criterio A: Complejidad

- [8] Complejidad constante
- [7] Complejidad logarítmica
- [6] Complejidad raíz
- [5] Complejidad lineal
- [4] Complejidad nlogn
- [3] Complejidad polinómica
- [2] Complejidad exponencial
- [1] Complejidad factorial

Criterio B: Eficiencia

- [3] Muy eficiente
- [2] Eficiente
- [1] Nada eficiente

Criterio C: Facilidad implementación

- [2] Fácil
- [1] Difícil

Evaluación

Evaluando los criterios anteriores en las alternativas que se mantienen, obtenemos la siguiente tabla:

	Criterio A	Criterio B	Criterio C	Total
Alternativa 3: Utilizar Pilas y Colas	Complejidad Constante - 8	Muy Eficiente - 3	Fácil - 2	13
Alternativa 4: Utilizar Hash Table	Complejidad Logarítmica – 7	Muy eficiente - 3	Fácil - 2	12
Alternativa 2: Utilizar Arreglos	Complejidad Lineal - 5	Eficiente – 2	Fácil - 2	9

Selección

De acuerdo con la evaluación anterior de se debe seleccionar las Alternativas 3 y 4, ya que obtuvieron la mayor puntuación de acuerdo a los criterios definidos.