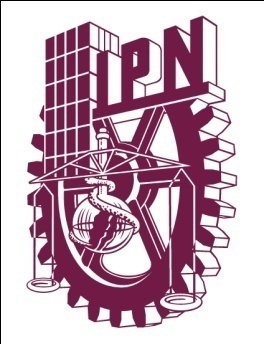
**INSTITUTO POLITÉCNICO NACIONAL**

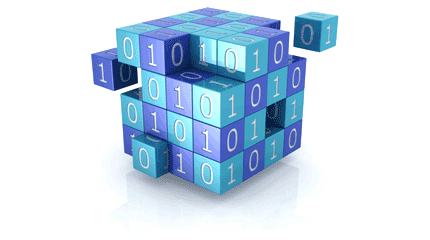
**ESCUELA SUPERIOR DE CÓMPUTO**

**ESTRUCTURAS DE DATOS**

PRÁCTICA #2

SIMULACIÓN CON EL TAD COLA

**GRUPO: 1CM8**

**EQUIPO: LAS MÁS PERRAS**

**INTEGRANTES:**

* JIMENEZ DELGADO LUIS DIEGO 2019630461
* SÁNCHEZ CASTRO AARÓN GAMALIEL 2019630079
* SANCHEZ TIRADO CITLALLI YASMIN 2019630096

**PROFESOR:** FRANCO MARTINEZ EDGARDO ADRIAN

**FECHA DE ENTREGA:** 11 DE MARZO 2019

INTRODUCCIÓN:

En el desarrollo de esta práctica se busca implementar una cola para representar los funcionamientos como es el de un banco, un supermercado o incluso la ejecución de procesos en el sistema

MARCO TEÓRICO:

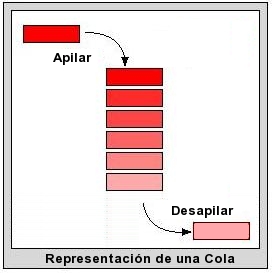
Concepto de cola: Una cola es una estructura de datos, caracterizada por ser una secuencia de elementos en la que la operación de inserción push se realiza por un extremo y la operación de extracción pop por el otro. También se le llama estructura FIFO (del inglés First In First Out), debido a que el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

Utilización

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.

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.



**Estados de una cola:**

Una cola puede estar:

Vacía: Se da siempre que la estructura no tiene elementos

Llena: Se da cuando no hay más lugar para almacenar los elementos.

Operaciones:

Queue: Agregar elemento; podemos agregar elementos siempre que la cola no esté llena y estos siempre los agregamos al final.

Dequeue: Podemos extraer el elemento insertado más antiguo, el cual se encuentra al frente siempre y cuando la cola no este vacia

Frente: Sin quitar el elemento podemos ver ele elemento que está próximo a salir.

**Errores detectados:**

Uno de los errores detectados fue que al momento de que usamos una librería como lo es “Windows.h” no compilaba en sistemas operativos como MAC OS

Conclusión de Yasmin:

Corroboro que una cola es un TDA ya que esta dedicado al almacenamiento y manipulación de los elementos, su funcionalidad siempre es la misma independientemente de la implementación que se haya utilizado.

Su funcionamiento cumple con la regla de FIFO que quiere decir que el orden de la salida de los elementos es el mismo que la entrada de los mismos.

Esto se debe a que las colas están diseñadas para devolver los elementos ordenados tal como llegan, para esto nos dimos cuenta que se puede decir que estas poseen un punto de acceso y otro de salida que lógicamente están ubicados en los extremos opuestos.

Mercadito:

En estás líneas de código podemos observar que corresponden a la declaración o bien llamado de las librerías que requerimos.

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

#include <time.h>

#include <string.h>

#include "TADColaDin.h"

#include "rep.h"

Aquí hacemos la declaración de constantes

#define ancho 90

#define repaintTime 20

#define tiempoEspaciado 200

#define espacioClientes 5

Está función nos sirve para ocultar el cursor.

void hidecursor**(**void**){**

HANDLE consoleHandle **=** GetStdHandle**(**STD\_OUTPUT\_HANDLE**);**

CONSOLE\_CURSOR\_INFO info**;**

info**.**dwSize **=** 100**;**

info**.**bVisible **=** FALSE**;**

SetConsoleCursorInfo**(**consoleHandle**,** **&**info**);**

**}**

En esta función se dibuja una caja que hace la simulación del supermecado

Recibe 2 enteros(x, y) que proporcionan las coordenadas en donde se comenzará a dibujar la caja; un entero (NumCaja)que guarda el numero de la caja que se dibujara; un carácter(estado), el cual describe el estado de la caja (si está abierta o cerrada); y finalmente un entero(velocidad), que almacena la velocidad en que la cajera atenderá los clientes

void dibujaCaja**(**int x**,** int y**,** int NumCaja**,** char estado**,** int velocidad**){**

MoverCursor**(**x**,**y**);**

printf**(**"\*\*\*\t"**);**

**if(**estado **==** 'A'**){**

MoverCursor**(**x**+**7**,** y**);**

printf**(**" "**);**

MoverCursor**(**x**+**7**,** y**);**

printf**(**"Atendiendo"**);**

**}**

**else**

**if(**estado **==** 'F'**){**

MoverCursor**(**x**+**7**,** y**);**

printf**(**" "**);**

MoverCursor**(**x**+**7**,** y**);**

printf**(**"CAJA CERRADA\n"**);**

**}**

**else{**

MoverCursor**(**x**+**7**,** y**);**

printf**(**" "**);**

MoverCursor**(**x**+**7**,** y**);**

printf**(**"Vacio"**);**

**}**

MoverCursor**(**x**+**2**,**y**+**1**);**

printf**(**"\*\t Caja %d"**,** NumCaja**+**1**);**

MoverCursor**(**x**,**y**+**2**);**

printf**(**"\*\*\*\t"**);**

printf**(**"Atendiendo 1 cliente/%d seg\n"**,** velocidad**);**

**}**

Función Cajeras:

Descripción: Dibuja todas las cajas que se indiquen, además imprime en pantalla: El estado de los clientes, el número de clientes que han sido atendidos, el nombre del supermercado y el estado del mercado (abierto o cerrado).

Recibe La cantidad de cajas que se van a dibujar; el espacio que tendrá la fila de cada cajera; el estado de cada una de las cajas, el nombre del supermercado, la velocidad de llegada de los clientes, la velocidad en que cada cajera atenderá a los clientes, la cantidad de clientes que han llegado y finalmente la cantidad de clientes atendidos.

Devuelve: un entero que representa una posición abajo de lo que dibujamos

int Cajeras**(**int cant**,** int largo**,** char estadoCajeras**[],** char marketName**[],** int llegadaClientes**,** int tiempoAtencion**[],** int cantClientesLlegados**,** int cantClientesAtendidos**){**

int i**;**

MoverCursor**(**0**,**0**);**

printf**(**" "**);**

MoverCursor**(**0**,**0**);**

printf**(**"%s \tRecibiendo 1 cliente/%d seg\tHan llegado %d clientes\t\t%d clientes han sido atendidos\t"**,** marketName**,** llegadaClientes**,** cantClientesLlegados**,** cantClientesAtendidos**);**

**if(**cantClientesAtendidos**>**100**){**

printf**(**"\t\tEL SUPERMERCADO CERRARA\n"**);**

**}**

**for(**i**=**0**;** i**<**cant**;** i**++){**

dibujaCaja**(**largo**+**1**,** i**\***4**+**1**,** i**,** estadoCajeras**[**i**],** tiempoAtencion**[**i**]);**

**}**

**return** i**\***4**+**5**;**

**}**

Función pintar:

Está función dibuja cada uno de los clientes del supermercado, recibe la fila en la que se dibujara, la posición, el cliente y un carácter que nos indicará si la fila ha terminado.

Para esto creamos un arreglo auxiliar de caracteres y luego en la siguiente línea hacemos una conversión a cadena el entero que esta dentro de nuestra estrucutura. Luego ubicamos las dos cordenadas (x,y) Hacemos una verificación para saber si ya no hay espacio en la fila y dependiendo el caso, se mueve el cursor.

void pintar**(**int filaPintar**,** int Posicion**,** elemento Cliente**,** char acaboFila**){**

char temp**[**4**];**

itoa**(**Cliente**.**n**,**temp**,**10**);**

int x **=** ancho **-** espacioClientes**\***Posicion **-** 3**;**

int y **=** filaPintar **\*** 4 **+** 2**;**

**if(**acaboFila **==** 'S'**){**

int i**;**

**for(** i**=**0**;** i**<**x**;** i**++){**

MoverCursor**(**i**,**y**);**

printf**(**" "**);**

**}**

**}**

**else{**

MoverCursor**(**x**,**y**);**

printf**(**" "**);**

MoverCursor**(**x**,**y**);**

fflush**(**stdout**);**

printf**(**"#%s"**,** temp**);**

**}**

**return;**

**}**

void limpiar**(**int i**){**

int y **=** i**\***4**+**2**;**

int c**;**

**for(**c**=**0**;** c**<**ancho**;** c**++)**

MoverCursor**(**0**,**y**);**

printf**(**"#"**);**

**return;**

**}**

Nuestra función principal se encarga de pedirle al usuario el nombre del supermercado, la cantidad de cajeras, cantidad de clientes, tiempo o bien velocidad en que serán atendidos, velocidad de llegada, así mismo también vamos llamando nuestras funciones para que comience el funcionamiento del super mercado.

int main**(**void**){**

unsigned int tiempo**,** cliente**,** atendidos**,** minClientes**,** fila**,**columna**,** alto**,** cantCajeras**,** filaElegida**,** llegadaClientes**;**

elemento aux**;**

char marketName**[**20**],** fin **=** 'N'**;**

printf**(**"Introduce el nombre del supermercado: "**);**

scanf**(**"%s"**,** **&**marketName**);**

printf**(**"Introduce la cantidad de cajeras (1-10): "**);**

scanf**(**"%d"**,** **&**cantCajeras**);**

**if(**cantCajeras**>**10 **||** cantCajeras **<** 1**){**

printf**(**"Cantidad de cajeras no valida\n"**);**

exit**(**0**);**

**}**

printf**(**"Introduce la minima cantidad de clientes para cerrar el supermercado: "**);**

scanf**(**"%d"**,** **&**minClientes**);**

char estadoCajeras**[**cantCajeras**];**

int tiempoAtencion**[**cantCajeras**];**

cola filaCajera**[**cantCajeras**];**

int i**;**

**for(** i**=**0**;** i**<**cantCajeras**;** i**++)**

Initialize**(&**filaCajera**[**i**]);**

**for(** i**=**0**;** i**<**cantCajeras**;** i**++){**

printf**(**"Tiempo de atencion de la caja #%d (multiplos de 10): "**,** i**+**1**);**

scanf**(**"%d"**,** **&**tiempoAtencion**[**i**]);**

**}**

printf**(**"Introduce la velocidad de llegada de Clientes: "**);**

scanf**(**"%d"**,** **&**llegadaClientes**);**

alto **=** Cajeras**(**cantCajeras**,** ancho**,** estadoCajeras**,** marketName**,** llegadaClientes**,** tiempoAtencion**,** cliente**,** atendidos**);**

tiempo **=** 0**;** cliente **=** 0**;** atendidos **=** 0**;**

system**(**"cls"**);**

hidecursor**();**

srand**(**time**(NULL));**

**while(**fin **==** 'N'**){**

tiempo**++;**

fin **=** 'S'**;**

int i**;**

**for(**i**=**0**;** i**<**cantCajeras**;** i**++){**

**if((!**Empty**(&**filaCajera**[**i**]))** **&&** estadoCajeras**[**i**]** **!=** 'F'**){**

**if(**tiempo **%** tiempoAtencion**[**i**]** **==** 0**){**

aux **=** Dequeue**(&**filaCajera**[**i**]);**

atendidos**++;**

**}**

estadoCajeras**[**i**]** **=** 'A'**;**

**}**

**else{**

estadoCajeras**[**i**]** **=** 'Z'**;**

**if(**atendidos**>**minClientes**)**

estadoCajeras**[**i**]** **=** 'F'**;**

**}**

Cajeras**(**cantCajeras**,**ancho**,** estadoCajeras**,** marketName**,** llegadaClientes**,** tiempoAtencion**,** cliente**,** atendidos**);**

**}**

**for(**int i**=**0**;** i**<**cantCajeras**;** i**++){**

**for(**int k**=**0**;** k**<=**Size**(&**filaCajera**[**i**]);** k**++){**

**if(**k **==** Size**(&**filaCajera**[**i**])){**

pintar**(**i**,**k**-**1**,**aux**,** 'S'**);**

**}**

**else{**

aux **=** Element**(&**filaCajera**[**i**],**k**+**1**);**

pintar**(**i**,** k**,** aux**,** 'N'**);**

**}**

**}**

**}**

**if((**tiempo **%** llegadaClientes **==** 0**)** **&&** atendidos**<=**minClientes**){**

cliente**++;**

aux**.**n **=** cliente**;**

filaElegida **=** rand**()%**cantCajeras**;**

Queue**(&**filaCajera**[**filaElegida**],** aux**);**

**}**

Sleep**(**repaintTime**);**

**for(**int a**=**0**;** a**<**cantCajeras**;** a**++){**

**if(**estadoCajeras**[**a**]** **!=** 'F'**)**

fin **=** 'N'**;**

MoverCursor**(**0**,** alto**+**5**);**

printf**(**"%c\n"**,** fin**);**

**}**

**}**

MoverCursor**(**0**,**alto**+**1**);**

**return** 0**;**

**}**