

Objetivo: Aplicar los conocimientos de colas en el lenguaje C.

Desarrollo:

**Ejercicio 1.** Una tienda de alimentos es atendida por una persona, la atención de los clientes se realiza siempre conforme su llegada.

La tienda considera 5 tipos de asuntos a atender:

1. Compra de alimentos.
2. Devolución.
3. Cambio.
4. Dudas.
5. Otros.

Utiliza una cola dinámica cuyos nodos representan a los clientes que quieren ser atendidos, cada cliente debe contar con un identificador único y con el asunto por el que va a la tienda.

Representa con un número aleatorio entre 20 y 100 la cantidad de clientes que llegarán a la tienda, que será la cantidad de nodos de la cola dinámica.

La tienda empieza a atender a las 8:00 am y termina de atender a las 16:00 pm. La atención de un cliente es equivalente a la acción de dequeue de la cola dinámica, cada que el cliente sea atendido genera de manera aleatoria los minutos que tardo en ser atendido, que este valor este entre 1 y 15.

La hora en que se empieza a atender el primer cliente será a las 8:00 am, para cada cliente que logre ser atendido imprime su identificador, el asunto por el que fue, la hora en que empezó a ser atendido y la hora final en ser atendido. Para los clientes que no alcanzaron a ser atendido solo imprime su identificador y un mensaje de que no fueron atendidos, y no olvides eliminar sus nodos.

**Pseudocodigo:**

Estructura Cliente

Estructura nodeDQ

Cliente e

nodeDQ siguiente

Estructura Dqueue

nodeDQ \*head

nodeDQ \*tail

Función createDQ(Dqueue \*dq)

dq->head = NULL

dq->tail = NULL

Función isEmpty(Dqueue \*dq) -> Booleano

Si dq->head == NULL entonces

Devolver Verdadero

Sino

Devolver Falso

Fin Si

Función enqueue(Dqueue \*dq, Cliente e)

nodeDQ \*nuevoNodo = CrearNodo(e)

Si isEmpty(dq) entonces

dq->head = nuevoNodo

Sino

dq->tail->siguiente = nuevoNodo

Fin Si

dq->tail = nuevoNodo

Función dequeue(Dqueue \*dq) -> Cliente

Cliente e

Si isEmpty(dq) entonces

Imprimir "Cola doble vacía"

Sino

nodeDQ \*temp = dq->head

e = temp->e

dq->head = dq->head->siguiente

LiberarMemoria(temp)

Devolver e

Fin Si

Función CrearNodo(Cliente e) -> nodeDQ \*

nodeDQ \*nuevoNodo = Crear nodoDQ

nuevoNodo->e = e

nuevoNodo->siguiente = NULL

Devolver nuevoNodo

Dqueue dq

createDQ(&dq)

enqueue(&dq, cliente1)

enqueue(&dq, cliente2)

cliente = dequeue(&dq)

**Código:**

Main.c

#include <stdio.h>

#include <stdlib.h>

#include "Cola.h"

void llenarCola(Dqueue \*, int);

void agregarCliente(Dqueue \*);

void atender(Dqueue \*, int);

void imprimirDatos(Cliente, int, int);

void main(){

    int cantClientes = rand() % 3;

    Dqueue cola;

    printf("Cantidad de clientes: %d\n", cantClientes);

    createDQ(&cola);

    llenarCola(&cola, cantClientes);

    atender(&cola, cantClientes);

}

void atender(Dqueue \*dq, int cantClientes){

    int minutos = 0;

    int hora = 8;

    int numClientes = cantClientes;

    while(hora <16 && !isEmpty(dq)){

        Cliente c;

        c = dequeue(dq);

        int minutosaAgregar = rand() % 81 + 20;

        if(minutos + minutosaAgregar >= 60){

            minutos = minutos + minutosaAgregar - 60;

            hora++;

        } else {

            minutos += minutosaAgregar;

        }

        imprimirDatos(c, hora, minutos);

        numClientes--;

    }

    if(numClientes > 0){

        printf("Quedaron %d clientes sin atender", numClientes);

        while(dq->head != NULL){

            Cliente c = dequeue(dq);

            printf("\nCliente %d\n", c.identificador);

        }

    }

}

void imprimirDatos(Cliente c, int hora, int minutos){

    printf("\nCliente %d\n", c.identificador);

    printf("Asunto: %d\n", c.asunto);

    printf("Hora de atencion: %d:%d\n", hora, minutos);

}

void llenarCola(Dqueue \*dq, int cantClientes){

    int i = 0;

    while(i < cantClientes){

        agregarCliente(dq);

        i++;

    }

}

void agregarCliente(Dqueue \*dq){

    Cliente nuevoCliente;

    printf("\n(1.- Compra de alimentos)\n(2.- Devolucion)\n(3.- Cambio)\n(4.-Dudas)\n(5.-Otro)\nSelecione el asunto: ");

    scanf("%d",&nuevoCliente.asunto);

    printf("\nIngrese su identificador(id numerico): ");

    scanf("%d",&nuevoCliente.identificador);

    enqueue(dq, nuevoCliente);

}

**Cola.c**

#include <stdio.h>

#include <stdlib.h>

#include "Cola.h"

void createDQ(Dqueue \*dq){

    dq->head = NULL;

    dq->tail = NULL;

}

int isEmpty(Dqueue \*dq){

    if(dq->head == NULL){

        return 1;

    } else {

        return 0;

    }

}

void enqueue(Dqueue \*dq, Cliente e){

    nodeDQ \*newNode = (nodeDQ\*)malloc(sizeof(nodeDQ));

    newNode->e = e;

    newNode->next = NULL;

    if(isEmpty(dq)){

        dq->head = newNode;

    } else {

        dq->tail->next = newNode;

    }

    dq->tail = newNode;

}

Cliente dequeue(Dqueue \*dq){

    Cliente e;

    if(isEmpty(dq)){

        printf("Empty\n");

    } else {

        nodeDQ \*temp = dq->head;

        e = temp->e;

        dq->head = dq->head->next;

        free(temp);

    }

    return e;

}

**Cola.h**

#ifndef \_Dqueue\_

#define \_Dqueue\_

typedef int element;

typedef struct{

    int asunto;

    int identificador;

}Cliente;

typedef struct nodeDQ{

    Cliente e;

    struct nodeDQ \*next;

}nodeDQ;

typedef struct Dqueue{

    nodeDQ \*head;

    nodeDQ \*tail;

}Dqueue;

void createDQ(Dqueue \*dq);

int isEmpty(Dqueue \*dq);

void enqueue(Dqueue \*dq, Cliente e);

Cliente dequeue(Dqueue \*dq);

#endif

**Resultados:**

Texto

Descripción generada automáticamente

**Ejercicio 2.** Realiza un programa que pida al usuario que introduzca una frase e indica si es una frase palíndromo o no. Utiliza una Pila y una cola estáticas para resolverlo.

**Pseudocodigo:**

Estructura squeue

Entero head

Entero tail

Entero array[Tamaño]

Estructura sstack

Entero top

Entero array[Tamaño]

Función create(squeue \*q)

q->head = 0

q->tail = -1

Función next(Entero index) -> Entero

Devolver (index + 1) % Tamaño

Función is\_empty(squeue \*q) -> Booleano

Si q->tail == -1 entonces

Devolver Verdadero

Sino

Devolver Falso

Fin Si

Función is\_full(squeue \*q) -> Booleano {

Si q->head == next(q->tail) y q->tail != -1 entonces

Devolver Verdadero

Sino

Devolver Falso

Fin Si

Procedimiento enqueue(squeue \*q, Entero x)

Si is\_full(q) entonces

Imprimir "La cola está llena"

Sino

q->array[next(q->tail)] = x

q->tail = next(q->tail)

Fin Si

Función dequeue(squeue \*q) -> Entero

Entero x

Si is\_empty(q) entonces

Imprimir "La cola está vacía"

Devolver -1

Sino

x = q->array[q->head]

Si q->head == q->tail entonces

q->head = 0

q->tail = -1

Sino

q->head = next(q->head)

Fin Si

Devolver x

Fin Si

Procedimiento Copiar(squeue \*q, squeue \*q2)

Entero i

Si is\_empty(q) entonces

Imprimir "La cola está vacía"

Sino

Para i = q->head hasta i != q->tail hacer

enqueue(q2, q->array[i])

Fin Para

enqueue(q2, q->array[q->tail])

Fin Si

Procedimiento mostrar(squeue \*q)

Entero i

Si is\_empty(q) entonces

Imprimir "La cola está vacía"

Sino

Para i = q->head hasta i != q->tail hacer

Imprimir q->array[i] + " "

Fin Para

Imprimir q->array[q->tail]

Fin Si

Procedimiento llenaCola(squeue \*q) {

Entero i = 0

Entero x

Imprimir "¿Desea ingresar un elemento a la cola? 1: Sí 0: No"

Leer x

Mientras x == 1 y i < Tamaño hacer

Imprimir "Ingrese el elemento: "

Leer x

enqueue(q, x)

Incrementar i

Imprimir "¿Desea ingresar otro elemento a la cola? 1: Sí 0: No"

Leer x

Fin Mientras

Si i == Tamaño entonces

Imprimir "La cola está llena"

Fin Si

Procedimiento createss(sstack \*s)

s->top = -1

Función isemptys(sstack \*s) -> Booleano

Si s->top == -1 entonces

Devolver Verdadero

Sino

Devolver Falso

Fin Si

Función isfulls(sstack \*s) -> Booleano

Si s->top == Tamaño - 1 entonces

Devolver Verdadero

Sino

Devolver Falso

Fin Si

Procedimiento pushs(sstack \*s, Entero e) {

Si isfulls(s) entonces

Imprimir "La pila está llena"

Sino

s->top++

s->array[s->top] = e

Fin Si

Función pops(sstack \*s) -> Entero

Si isemptys(s) entonces

Imprimir "La pila está vacía"

Sino

s->top--

Devolver s->array[s->top + 1]

Fin Si

Procedimiento mostrarPil(sstack \*s)

Entero i

Para i = 0 hasta i <= s->top hacer

Imprimir s->array[i]

Fin Para

Imprimir una línea nueva

**Código:**

Palíndromo.c

#include <stdio.h>

#include <stdlib.h>

#include "Palindromo.h"

#include <ctype.h>

void palindromo(squeue \*, sstack \*);

void main(){

    char A[50];

    printf("Ingrese la palabra: ");

    scanf("%s", A);

    printf("\n");

    sstack pila;

    createss(&pila);

    squeue cola;

    create(&cola);

    char b;

    for(int i = 0; A[i] != '\0'; i++){

        b = tolower(A[i]);

        enqueue(&cola, b);

        pushs(&pila, b);

    }

    palindromo(&cola, &pila);

}

void palindromo(squeue \*q, sstack \*s){

    char a, b;

    int a1,b1;

    while(!isemptys(s) && !is\_empty(q)){

        a = dequeue(q);

        b = pops(s);

        if (a != b){

            printf("\nNo es palindromo\n");

            return;

        }

    }

    a1 = is\_empty(q);

    b1 = isemptys(s);

    if(a1 == 0 && b1 == 0){

        printf("\nEs palindromo\n");

    }

    if(a1 == 1 && b1 == 1){

        printf("\nEs palindromo\n");

    }

}

funcionesPal.c

#include <stdio.h>

#include <stdlib.h>

#include "Palindromo.h"

void create(squeue \*q){

    q->head = 0;

    q->tail = -1;

}

int next(int index){

    return (index + 1) % size;

}

int is\_empty(squeue \*q){

    if(q->tail == -1){

        return true;

    }else{

        return false;

    }

}

int is\_full(squeue \*q){

    if(q->head == next(q->tail) && q->tail != -1){

        return true;

    }else{

        return false;

    }

}

void enqueue(squeue \*q, element x){

    if(is\_full(q)){

        printf("Queue is full\n");

    }else{

        q->array[next(q->tail)] = x;

        q->tail = next(q->tail);

    }

}

int dequeue(squeue \*q){

    element x;

    if(is\_empty(q)){

        printf("Queue is empty\n");

        return -1;

    }else {

        x = q->array[q->head];

        if(q->head == q->tail){

            q->head = 0;

            q->tail = -1;

        }else{

            q->head = next(q->head);

        }

        return x;

    }

}

void Copiar(squeue \*q, squeue \*q2){

    int i;

    if(is\_empty(q)){

        printf("Es vacia\n");

    }else{

        for(i = q->head; i != q->tail; i = next(i)){

            enqueue(q2, q->array[i]);

        }

        enqueue(q2, q->array[q->tail]);

    }

}

void mostrar(squeue \*q){

    int i;

    if(is\_empty(q)){

        printf("Es vacia\n");

    }else{

        for(i = q->head; i != q->tail; i = next(i)){

            printf("%d ", q->array[i]);

        }

        printf("%d\n", q->array[q->tail]);

    }

}

void llenaCola(squeue \*q){

    int i = 0;

    int x;

    printf("Desea ingresar un elemento a la cola 1:SI 0:NO\n");

    scanf("%d", &x);

    while(x == 1 && i < 10){

        printf("Ingrese el elemento: ");

        scanf("%d", &x);

        enqueue(q, x);

        i++;

        printf("Desea ingresar un elemento a la cola 1:SI 0:NO\n");

        scanf("%d", &x);

    }

    if (i == 10){

        printf("Cola llena\n");

    }

}

void createss(sstack \*s) {

    s->top = -1;

}

int isemptys(sstack \*s) {

    if(s->top == -1) {

        return true;

    } else {

        return false;

    }

}

int isfulls(sstack \*s) {

    if(s->top == size - 1) {

        return true;

    } else {

        return false;

    }

}

void pushs(sstack \*s, element e) {

    if(isfulls(s)) {

        printf("Stack is full\n");

    } else {

        s->top++;

        s->array[s->top] = e;

    }

}

int pops(sstack \*s) {

    if(isemptys(s)) {

        printf("Stack is empty\n");

    } else {

        s->top--;

        return s->array[s->top + 1];

    }

}

void mostrarPil(sstack \*s) {

    int i;

    for(i = 0; i <= s->top; i++) {

        printf("%d\n", s->array[i]);

    }

    printf("\n");

}

Palindromo.h

#ifndef \_palin\_

#define \_palin\_

#define size 50

typedef char element;

#define true 1

#define false 0

typedef struct Queue{

    int head;

    int tail;

    element array[size];

} squeue;

typedef struct Stack {

    element array[size];

    int top;

} sstack;

void create(squeue \*q);

int next(int index);

int is\_empty(squeue \*q);

int is\_full(squeue \*q);

void enqueue(squeue \*q, element x);

int dequeue(squeue \*q);

void mostrar(squeue \*q);

void Copiar(squeue \*q, squeue \*q2);

void llenaCola(squeue \*q);

void createss(sstack \*s);

int isemptys(sstack \*s);

int isfulls(sstack \*s);

void pushs(sstack \*s, element e);

int pops(sstack \*s);

void mostrarPil(sstack \*s);

#endif

Resultado:

Texto

Descripción generada automáticamente

**Conclusiones:**

*Rivera Plascencia Bryan*

¿Cuál fue su experiencia con la practica?

Fue un poco complejo desarrollar el ejercicio 1 pero al final si se pudo resolver de manera correcta.

Si tuvieron errores ¿Cómo lo solucionaron?

Fue el uso de los apuntadores, haciendo uso del “&”.

Si aprendieron algo nuevo ¿Qué fue?

Como implementar la pila y la cola al mismo tiempo

*Velazquez Trinidad Jazmin Montserrath*

¿Cuál fue su experiencia con la practica?

Personalmente considero que el primer ejercicio lo senti un poco pesado por pensar todos los nodos que se tenían que aplicar

Si tuvieron errores ¿Cómo lo solucionaron?

Como en la practica anterior el implementar los apuntadores; sin embargo fue mas fácil ya que siento que nos ayudo justo la practica pasada

Si aprendieron algo nuevo ¿Qué fue?

Aprender a hacer uso tanto pila como cola juntos

**Bibliografía:**

* (S/f). Wordpress.com. Recuperado el 1 de mayo de 2024, de <https://utch2009.wordpress.com/wp-content/uploads/2010/04/apuntadores.pdf>
* TylerMSFT. (s/f). Punteros (C++). Microsoft.com. Recuperado el 1 de mayo de 2024, de <https://learn.microsoft.com/es-es/cpp/cpp/pointers-cpp?view=msvc-170>