UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA

Algoritmos y estructura de datos

Practica 9. Estructura FIFO

Alumno: Caudillo Sánchez Diego

Matricula: 1249199

Grupo: 551

Docente: Alma Leticia Palacios Guerrero

Fecha de entrega: 10/Mayo/2019

Introducción

Una cola es una estructura de datos que almacena elementos en una lista y permita acceder a los datos por uno de los dos extremos de la lista. Un elemento se inserta en la cola (parte final) de la lista y suprime o elimina por la frente (parte inicial, cabeza) de la lista. Las aplicaciones utilizan una cola para almacenar elemento en su orden de aparición o concurrencia.

Los elementos se eliminan (se quitan) de la cola en el mismo orden en que se almacenan y, por consiguiente, una cola es una estructura de tipo FIFO (*First In-First Out*). el servicio de atención a cliente de un almacén es un ejemplo típico de cola. La acción de gestión de memoria intermedia (*buffering*) de trabajos o tareas de impresora en un distribuidor de impresoras (*spooler*) es otro ejemplo típico de cola. Dado que la impresión es una tarea que requiere mas tiempo que el proceso de la transmisión real de los datos desde la computadora a la impresora, se organiza una cola de trabajos de modo que los trabajos se imprimen en el mismo orden en que se recibieron por la impresora. Este sistema tiene el gran inconveniente de que, si su trabajo personal consta una única página para imprimir y delante de su petición de impresión existe otra petición para imprimir un informe de 300 páginas, deberá esperar a la impresión de esas 300 paginas antes de que se imprima su página.

Desde el punto de vista de estructura de datos , una cola es similar a una pila, en donde los datos se almacenan de un modo línea y el acceso a los datos solo esta permitido en los extremos de la cola. las acciones que están permitidas en una cola son:

- Creación de una cola vacía.
- Verificación de que una cola está vacía.
- Añadir un dato al final de una cola.
- Eliminación de los datos de la cabeza de la cola.

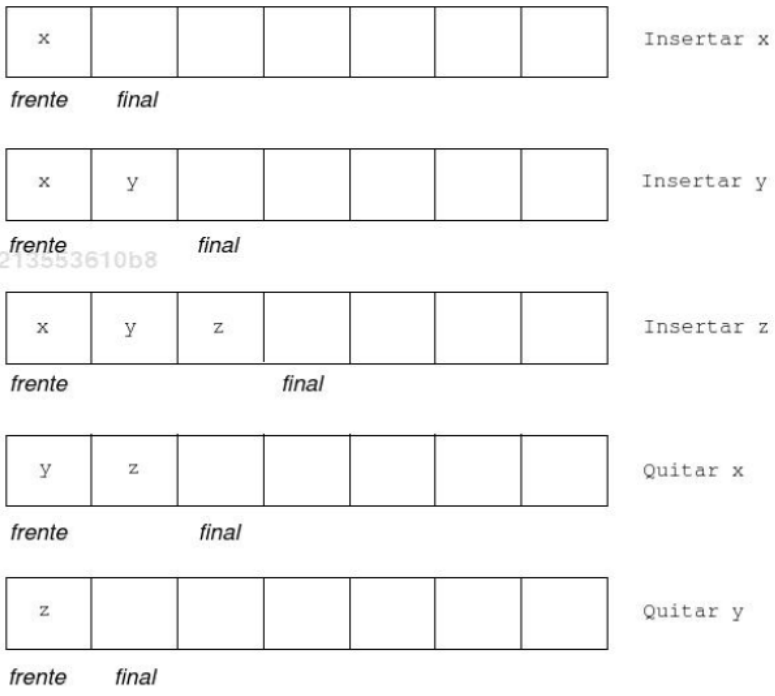

Figura. Operaciones de *Insertar* y *Quitar*

Competencia
Implementar soluciones creativas de software utilizando eficientemente el principio FIFO.

Problema
Desarrolle un programa que permita dar entrada a polinomios de x, representándolos con una cola. A continuación, obtener una tabla de valores del polinomio para valores de x=0.0, 0.5, 1.0, 1.5,.5.0
El programa debe tener las siguientes opciones:

1) Capturar término.
2) Ver polinomio.
3) Generar tabla de valores.
4) Salir.

Código

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

struct nodo
{
    int exponente;
    float valor;
    struct nodo* next;
};

void menu(void);
struct nodo* createQueue(float value, int exp);
struct nodo* insert(struct nodo* inicio, float value, int exp);
void mostrar(struct nodo* inicio);
void generateValueTable(struct nodo* queue);

int main(int argc, char const *argv[])
{
    int x,y;
    menu();
    return 0;
}

void menu(void)
{
    int opc, exponente;
    float value;
    struct nodo* queue = NULL;
    do
    {
```

```
        system("cls");
        printf("[1] Capturar t%crmino.\n",130);
        puts("[2] Ver polinomio.");
        puts("[3] Generar tabla de valores.");
        puts("[4] Salir.");
        printf("Opci%cn [ ]\b\b",162);
        scanf(" %d", &opc);

        switch (opc)
        {
            case 1: printf("Capture la funci%cn a evaluar. Ejemplo: Termino--
>[2]X%c[3]<--Exponente\n",162,94);
                printf("Termino: "); scanf("%f", &value);
                printf("Exponente: "); scanf("%d", &exponente);
                if(!queue) queue = createQueue(value, exponente);
                else queue = insert(queue, value, exponente);
                break;
            case 2: puts(">>>Ver polinomio<<<");
                mostrar(queue);
                break;
            case 3: puts(">>>Generar tabla de valores<<<");
                generateValueTable(queue);
                setbuf(stdin, NULL);
                getchar();
                break;
        }
    } while (opc != 4);
    free(queue);
}

struct nodo* createQueue(float value, int exp)
{
    struct nodo* queue = (struct nodo*) malloc(sizeof(struct nodo));
    if(!queue)
    {
        printf("No fue posible la asignaci%cn de memoria a la cola\n",162);
        free(queue);
        exit(EXIT_FAILURE);
    }
    queue->valor = value;
    queue->exponente = exp;
    queue->next = NULL;
    return (queue);
}
```

```c
struct nodo* insert(struct nodo* inicio, float value, int exp)
{
    struct nodo *p,*q, *nuevo;
    nuevo = createQueue(value, exp);
    p = inicio;
    while(p != NULL){
        q = p;
        p = p->next;
    } q->next = nuevo;
    return inicio;
}

void mostrar(struct nodo* inicio)
{
    struct nodo* aux;
    system("cls");
    if(!inicio) printf("Vacia.\n");
    else
    {
        puts(">>>Polinomio<<<");
        aux = inicio;
        do
        {
            if(aux->valor >= 0) printf("+ ");
            printf("%.1fX%c%d ", aux->valor, 94 ,aux->exponente);
            aux = aux->next;
        } while (aux);
        setbuf(stdin, NULL);
        getchar();
    }
}

void generateValueTable(struct nodo* queue)
{
    struct nodo* aux;
    float x;
    system("cls");
    if(!queue) puts("Cola se encuentra vacia.");
    else
    {
        aux = queue;
        do
        {
            printf("\nFunci%cn %.1fx%c%d\n", 162, aux->valor, 94 ,aux->exponente);
            for(int i = 0; i <= 10; i++)
```

```
            {
                printf("f(%.2f) = %.2f\n", x, pow((aux->valor*x), aux->exponente));
                x += 0.5;
            }
            x = 0;
            aux = aux->next;
        } while (aux);
    }
}
```

Evidencia de ejecución

```
[1] Capturar término.
[2] Ver polinomio.
[3] Generar tabla de valores.
[4] Salir.
Opción [1]
Capture la función a evaluar. Ejemplo: Termino-->[2]X^[3]<--Exponente
Termino: 5
Exponente: 3
```

```
>>>Polinomio<<<
+ 5.0X^3 + 6.0X^2 -6.0X^1
```

```
Función 2.0x^3
f(0.00) = 0.00
f(0.50) = 1.00
f(1.00) = 8.00
f(1.50) = 27.00
f(2.00) = 64.00
f(2.50) = 125.00
f(3.00) = 216.00
f(3.50) = 343.00
f(4.00) = 512.00
f(4.50) = 729.00
f(5.00) = 1000.00
```

```
Función -5.0x^3
f(0.00) = -0.00
f(0.50) = -15.63
f(1.00) = -125.00
f(1.50) = -421.88
f(2.00) = -1000.00
f(2.50) = -1953.13
f(3.00) = -3375.00
f(3.50) = -5359.38
f(4.00) = -8000.00
f(4.50) = -11390.63
f(5.00) = -15625.00
```

```
Función 6.0x^0
f(0.00) = 1.00
f(0.50) = 1.00
f(1.00) = 1.00
f(1.50) = 1.00
f(2.00) = 1.00
f(2.50) = 1.00
f(3.00) = 1.00
f(3.50) = 1.00
f(4.00) = 1.00
f(4.50) = 1.00
f(5.00) = 1.00
```

```
Función 10.0x^1
f(0.00) = 0.00
f(0.50) = 5.00
f(1.00) = 10.00
f(1.50) = 15.00
f(2.00) = 20.00
f(2.50) = 25.00
f(3.00) = 30.00
f(3.50) = 35.00
f(4.00) = 40.00
f(4.50) = 45.00
f(5.00) = 50.00
```

Conclusión

Como se mencionaba al inicio del trabajo, las colas difieren en las pilas en varios puntos, de los mas importantes se encuentra en la forma que los datos se extraen y como se insertan. Una de las ventajas de la cola es el fácil manejo de su estructura, aunque sea muy parecido a las pilas, al momento de quitar un dato este no lo saca directamente de la cola, lo único que hace es recorrer ese dato dejándolo pasar. A esto le podemos notar una desventaja ya que esa locación de memoria se desperdicia ya que el dato ahí se queda, mas no sé utiliza. El uso de colas es muy útil en distintos aspectos, tal cual lo miramos en la práctica concluida. La cual permite evaluar funciones que el usuario ingrese. El programa esta creado para evaluarlo con números constantes, pero con una modificación al código pudiesen evaluarse con cualquier valor que el usuario desee. Esta clase de problemas ayudan a entender de mejor manera el concepto de colas y las posibles utilizaciones que se les puede dar. Ya a partir de esto nos damos cuenta las diferentes áreas en que las colas están presentes.

Bibliografía

Joyanes L. & Zahonero I. (2008). Colas y pilas. *En Metodología, algoritmos y estructura de datos. (*641-649). Madrid: McGraw Hill.