Trabalho de Estrutura de Dados e Mobile

By

Filipi de Luca Valim dos Santos
RA 22216027-7
Gabriel Diniz Gisoldo
RA 22214007-1

Sumário

1.0 Pseudocódigo

1.1 Fila Estática Circular

```
Início do Algoritmo

    Arrow Função LDDE:

    Início da Arrow Função
Var
    vetor
    vetorAuxiliar,
    início <- 0,
    fim <- 0,
    tamanho <- 5, // para um fila de tamanho 5
    AuxiliarRemove <- 0

        Arrow Funcao Adiciona

        Início da Arrow Funcao (valor: inteiro): Boleano

            SE < <fim + 1> % <tamanho + 1> == início> ENTAO
                    retorne falso
            FIMSE

            SENAO
                vetor[fim] <- valor
                fim <- (fim + 1) % (tamanho + 1)
                retorne verdadeiro
            FIMSENAO

        Fim da Arrow Funcao



        Arrow Funcao AuxiliarRemove

            Inicio Arrow Funcao ()

            SE < fim == 0 > ENTAO
                retorne falso
            FIMSE

            SENAO SE < tamanho do vetor == 1>ENTAO
                Vetor[0] <- null

            FIMSENAO

            PARA contador de 1 ATE tamano do vetor FACA

                vetorAuxiliar[AuxiliarRemove] <- Vetor[contador]

                AuxiliarRemove Incrementa

            FIMPARA

            AuxiliarRemove <- 0

            PARA contador de 1 ATE tamano do vetor FACA

                vetor[contador] <- vetorAuxiliar[AuxiliarRemove]

                AuxiliarRemove Incrementa

            FIMPARA

        Fim da Arrow Funcao
```

```
        Arrow Funcao Busca

        Inicio Arrow Funcao (valor: inteiro): inteiro

            PARA contador de 0 ATE tamanho do vetor FACA

                SE < vetor[contador] == valor ENTAO
                    retorne escreva(vetor[contador])
                FIMSE

                SENAO
                    retorne escreva(valor nao encontrado)
                FIMSENAO

            FIMPARA

        Fim da Arrow Funcao

    Fim da Arrow Funcao


Fim do Algoritmo
```

1.2 – LDDE

```
Inicio do Algoritmo

    Funcao Ldde
        Var
            Head <- null
            last <- null
            tamanho <- null



            Arrow Funcao Node

                retorne valor, ProximoNo <- null, UltimoNo <- null

            Fim Arrow Funcao


            Arrow Funcao Adiciona(valor: inteiro)

                SE <Head != null> ENTAO

                    Head <- [Obejeto Node(valor)]
                    last <- Head
                    incrementa tamanho

                FIMSE

                var newNo = [Obejeto Node(valor)]
                last.ProximoNo <-Head
                newNo <- UltimoNo
                UltimoNo <-newN
                retorne newNo

            FIMFUNCAO

            Arrow Funcao Remove(valor: inteiro)inteiro

                SE<tamanho == 0> ENTAO
                    retorne
                FIMSE

                SENAO SE < valor == Head> ENTAO
                    Head <- valor.ProximoNo
                    retorne verdadeiro
                FIMNAOSE

                var auxilarNo <- Head

                ENQUANTO <auxilarNo.newNo && auxilarNo.newNo != valor> FACA
                   auxilarNo <- auxilarNo.newNo

                FIMENQUANTO

                auxilarNo.newNo = valor.newNo

                retorne verdadeiro

            FIMFUNCAO
```

```
        Arrow Funcao Busca(valor: inteiro):inteiro

            var node <- Head

            SE< tamanho == 0> FACA
                retorne Nulo
            FIMSE

            SENAOSE < valor do nó == valor> FACA
                retorne noede
            FIMNAOSE

            ENQUANTO <node.ProximoNo != null> FACA
                node <- node.newNo
                    SE <valor do nó == valor digitado>FACA
                        retorne nó
                    FIMSE
            FIMENQUANTO

            retorne nó

        Fim da Arrow Funcao
Fim do Algoritmo
```

2.0 Análise e descrição dos Algoritmos

2.1 Fila Estática Circular

Filas Estática Circular são uma estrutura para organizar os dados, seu princípio consistem em “FIFO” do inglês: First In, First Out, primeiro dado que entra é o primeiro que sai ou seja a ordem de chegada dos dados importa, sendo assim usando um exemplo da vida real, a Fila onde o primeiro que chega é o primeiro que é atendido.

O algoritmo implementado na linguagem JavaScript neste trabalho, usa “Arrow Funciton” principal chamada: `Dados_Fila = () => {` onde dentro dela há mais 4 “Arrow Function”, são elas `const add = (value) =>`, `const remove = () => {`, `const Busca = (value) => {` e `const print = () =>` que respectivamente adicionam, removem, buscam e imprimem a lista.

Descrevendo a Função: `const add = (value) =>`

```
const add = (value) => {

    if ((fim + 1) % (tamanho + 1) == inicio) {
        return false;
    }

    vetor[fim] = value;
    fim = (fim + 1) % (tamanho + 1);

    return true;

}
```

Na Arrow Function “add”, irá ser recebido um “value” que corresponde ao valor digitado pelo usuário que será inserido na fila, logo após no primeiro “if” é verificado se o fim da fila pelo

resto da divisão com o tamanho da fila é igual ao inicio da fila ou seja isto é feito para verificar se toda fila já foi preenchida e verificar se poderá ser inserido um novo número, se isso ocorrer irá retornar um “false”, o que ira parar o programa e não deixará inserir novos dados. Caso o “if” seja falso, ou seja, ainda houver espaço para a inserção ele irá inserir no último espaço livre da fila e em seguida realizar o cálculo para incrementar o a variável “fim” para a próxima inserção.

Descrevendo a Função: `const remove = () => {`

```
const remove = () => {


    if (fim == 0) {
        return false
    }

    if (vetor.length == 1) {

        vetor[0] = null
    }

    for (let i = 1; i < vetor.length; i++) {
        vetorAux[auxremove] = vetor[i]
        auxremove++;
    }

    auxremove = 0
    for (let i = 0; i < vetorAux.length; i++) {
        vetor[i] = null
        vetor[i] = vetorAux[auxremove]
        auxremove++

    }

    fim--
}
```

Na Arrow Function “remove” ela não receberá nenhum valor de parâmetro pois ela irá remover o primeiro valor da Fila, pois a Fila Circular Estática funciona no princípio FIFO, ou seja, primeiro dado que entra é o primeiro que sai.

No primeiro “if” será verificado se o fim da fila é igual a 0 ou seja isso indica que a fila esta vazia, caso contrário será verificado no segundo “if” se a fila tem tamanho 1, pois se isso for verdadeiro será atribuído “null” ao único dado da fila, o que não precisara ser executado o passo seguinte.

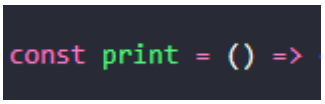
Caso a todas as condições dos “ifs” forem falsas será executado um comando de repetição que para armazenar todos os valores em um vetor com uma posição a menos e em um próximo “loop” ele irá copiar primeiro em todas as posições o valor “null”, e após isso ele irá copiar todas as posições do vetor auxiliar para o vetor original e assim terá todas as posições menos a última que será “null”.

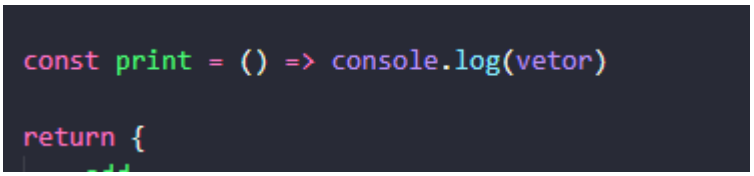
Descrevendo a Função: `const Busca = (value) => {`

```
const Busca = (value) => {

    for (let i = 0; i < vetor.length; i++) {
        if (vetor[i] == value) {
            return console.log(vetor[i])
        } else return console.log("Valor não encontrado")


    }

}
```

Na Arrow Function Busca ele irá receber um valor que será verificado sua existência na fila, no primeiro e único “loop”, dentro do mesmo será verificado todas as posições a correspondência do valor digitado pelo usuário, caso encontrado

irá retornar o valor, caso contrário será retornado uma mensagem “Valor não encontrado”.

Descrevendo a Função: 



A Arrow Function “print” é apenas para imprimir o vetor que representa a estrutura Fila.

2.2 LDDE - (Lista Dinâmica Duplamente Encadeada)

LDDE é outra estrutura usada para armazenar dados, seu princípio consiste em criar “nós” que armazenam um número e neste mesmo armazenam o endereço do próximo e do último “nó”, permitindo que seja armazenado alocada e não continua na memória.

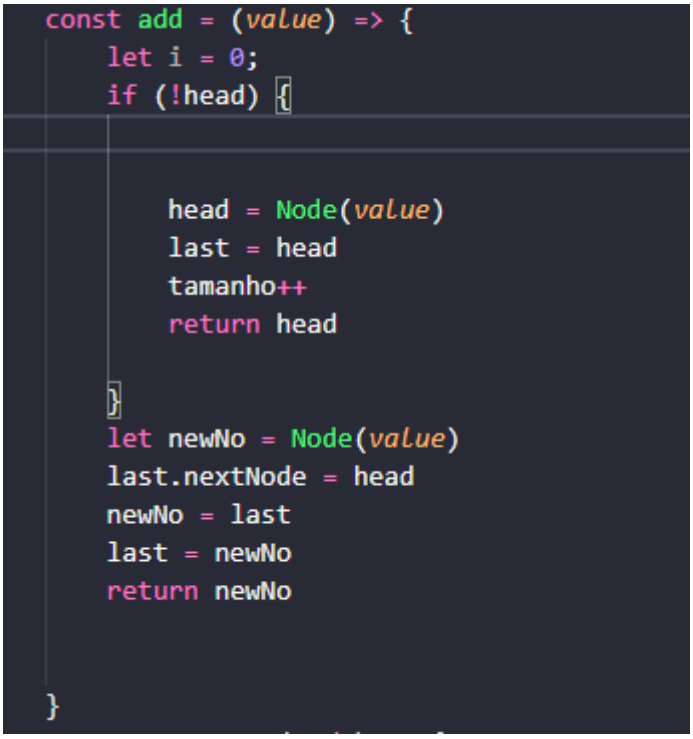
O algoritmo implementado na linguagem JavaScript neste trabalho, usa função chamada: `function Ldde() {` onde dentro dela há 4 “Arrow Function”, são elas `const Node = (value) => {` `const add = (value) =>`, `const remove = () => {`, `const Busca = (value) => {` e que respectivamente o NÓS da estrutura, adicionam, removem e buscam na estrutura.

Descrevendo a Função: `const Node = (value) => {`

```
const Node = (value) => {
    return {
        value,
        nextNode: null,
        lastNode: null
    }
}
```

Na Arrow Function “Node” ela irá funcionar para criar o “nó” que inicialmente o “próximo nó e o último nó” será “null”.

Descrevendo a Função: `const add = (value) => {`



```
const add = (value) => {
    let i = 0;
    if (!head) {

        head = Node(value)
        last = head
        tamanho++
        return head

    }
    let newNo = Node(value)
    last.nextNode = head
    newNo = last
    last = newNo
    return newNo


}
```

Na Arrow Function “add” será criado nós e será inserido valores nessa estrutura, ela irá receber um valor e verificará no primeiro “if” se a chama “cabeça” que significa o primeiro nó da estrutura, se a mesma esta diferente de “null”, se não será inserido o primeiro valor na LDDE e o próximo nó será “null” até a próxima inserção e o ultimo nó será ele próprio pois este é o primeiro da estrutura. Caso o primeiro “if” seja considerado falso, será criado um nó, mas neste caso ele irá guardar o enderenço do último e não mais dele mesmo pois este não é a cabeça da estrutura.

Descrevendo a Função: `const remove = (node) => {`

```
const remove = (node) => {

    if (tamanho == 0) return

    else if (node == head) {
        head = node.nextNode
        return true
    }


    let currentNode = head
    while (currentNode.nextNode && currentNode.nextNode != node) {

        currentNode = currentNode.nextNode


    }
    currentNode.nextNode = node.nextNode
    return true


}
```

Na Arrow Function “remove” será removido um nó e consequentemente um valor, sendo assim ela irá receber um valor, que no caso é o valor que o usuário deseja remover. No primeiro “if” será verificado se o tamanho da estrutura é igual a zero, esta verificação ocorre, pois, caso isto seja verdadeiro o algoritmo irá parar e não será removido nada pois a LDDE está vazia, caso seja falso esta situação ele entrar em um “loop” onde ele percorrera todo estrutura até chegar na última e após isso ele irá inserir o novo nó.

Descrevendo a Função: `const Busca = (index) => {`

```
const Busca = (index) => {
    let node = head

    console.log(tamanho)
    if (tamanho == 0) {
        return null
    } else if (node.value == index) {
        return node
    }


    while (node.nextNode) {
        node = node.nextNode
        if (node.value == index) {
            return node
        }
    }
    return node
}

return {
```

Na Arrow Function “Busca” ela irá retornar o nó como valor desejado da busca, a função inicia recebendo um valor em seguida ele guardara o valor da cabeça em uma variável auxiliar, após isso ele verificara no primeiro “if” se a estrutura esta vazia, caso positivo será retornado “null” pois não há valores para retornar, caso negativo será testado no próximo “if” se o primeiro nó é o procurado se sim ele retorna o mesmo caso contrário ele entrara em um “loop” que irá percorrer a estrutura procurando o valor e irá encontrar o mesmo, caso contrário será retornado “null” pois o valor procurado não se encontra na LDDE.