# Trabalho Final da Disciplina Algoritmos e Estrutura de Dados I
# TAD Deque

**Flávio E. O. e Silva, Isabela de S. Silva, Maycon P. Goulart, Sandro R. dos Reis**

[1]Universidade Federal de Itajubá (UNIFEI)
CEP – 37.500-903 – Itajubá – MG – Brasil

{flaedu99, souza.isa96, mayconpgoulart}@gmail.com, sandrouh@hotmail.com

***Abstract.*** *This work aims to demonstrate the implementation of a data structure called deque, the same thing treated as a special type of queue. While working, you can view a feature of the deck and some of its applications. In addition to understanding an implementation of itself.*

***Resumo.*** *Este trabalho tem como objetivo demonstrar a implementação de uma estrutura de dados denominada deque, a mesma será tratada como um tipo especial de fila. Ao decorrer do trabalho, será possível compreender a funcionalidade do deque, e algumas de suas aplicações. Além de entender a implementação tanto estática, quanto dinâmica, do mesmo.*

## 1. Introdução

A estrutura de dados deque (Double Ended Queue) trata-se de um tipo especial de fila duplamente encadeada. Em uma fila todas as inserções são realizadas em um extremo, e as remoções ocorrem em seu extremo oposto. Já no deque, não há essa regra e tanto a inserção, quanto a remoção, podem ser realizadas em ambas extremidades da estrutura.

Apesar de sua semelhança com uma fila, a estrutura deque generaliza a ideia de fila e pilha, podendo ser utilizado como substituto de ambos. Na estrutura pilha, e devido suas particularidades, a inserção e remoção ocorrem no mesmo extremo.

## 2. Descrição da Implementação

A estrutura deque pode ser implementada de duas formas, estáticamente e dinamicamente. A seguir, estará listado e explicado as funções utilizadas nas duas implementações:

### 2.1. Função Cria Deque

```
deque* cria_deque() {
deque *d = (deque*)malloc(sizeof(deque));
if (!d){
        printf(”Erro ao alocar”);
        exit(1);
}
d→final = NULL;
d→inicio = NULL;
```

```
d→qtd = 0;
return d;
}
```
A função Cria Deque, irá alocar um espaço de memória reservado para a estrutura. A única diferença entre a implementação Dinâmica e da Estática, é o fato de que na Dinâmica, o final e o início inicializarão em NULL, para mostrar que não há nenhum dado, e no caso da Estática, inicializará em 0.

### 2.2. Função Insere Início

```
void insere_inicio(deque *d, int i) {
        if (!d)
        return ;
}
no *novono = (no*)malloc(sizeof(no));
        if (!novono) {
        printf("Erro ao alocar");
        exit (1);
}
novono→dados.num = i;
novono→prox = d→inicio;
novono→ant = NULL;
if (!d→inicio)
        d→final = novono;
else
        d→inicio→ant = novono;
d→inicio = novono;
d→qtd++;
return ;
}
```
A função Insere Início, irá inserir um número, digitado pelo usuário, na 1ª posição do deque. Dinamicamente, ele irá alocar um novo nó e verificará se há memória suficiente. Feito isso, ele irá inserir o número no nó, apontará o novono próximo para o antigo início e o seu anterior para NULL, pois ficará na 1ª posição. Por último, verificará se a lista estava vazia ou não. Caso esteja, o final também será o novono, senão, fará o antigo início apontar para o novono, a fim de manter duplamente encadeado e incrementará em 1 a quantidade.

```
if (d→qtd == MAX) {
        printf("O Deque está cheio, não foi possível inserir");
        setbuf(stdin, NULL);
        getchar();
        return ;
```

```
}
d→inicio - -;
if (d→inicio ≺ 0)
        d→inicio = MAX-1;
```

A única diferença para o Estático, é que ele verificará se já estourou o limite do deque, caso estoure, não poderá inserir, senão, recuará o início em uma posição e irá inserir em seu lugar. Caso o início fique negativo, ele voltará para a posição MAX-1 (MAX é o tamanho máximo pré-determinado), para manter o caráter circular do deque.

### 2.3. Função Insere no Final

```
void insere_final(deque *d, int i){
if (!d)
        return ; no *novono = (no*)malloc(sizeof(no));
if (!novono) {
        printf("Erro ao alocar");
        exit (1);
        novono→dados.num = i;
        novono→prox = NULL;
if (!d→final) {
        novono→ant = NULL;
        d→inicio = novono;
}
else {
        novono→ant = d→final;
        d→final→prox = novono;
}
d→final = novono;
d→qtd++;
return ;
}
```

A função Insere Final irá inserir um número, digitado pelo usuário, na última posição do deque. Dinamicamente, ele irá alocar um novo nó e verificará se há memória suficiente. Feito isso, ele irá inserir o número no nó, apontará o novono próximo para NULL, por ser o último elemento. Irá verificar se possui apenas um elemento na lista, caso haja, apontará o seu anterior para NULL também e o início do deque irá ser o novono. Senão, o novono anterior apontará para o antigo final, para se manter duplamente encadeado e o próximo do final anterior, apontará para o novono. Por fim, o novono assumirá o final e será incrementado em 1 a quantidade.

```
        if (d→qtd == MAX) {
        printf("O Deque está cheio, não foi possível inserir");
```

```
        setbuf(stdin, NULL);
        getchar();
        return ;
}
d→dados[d→final].num = i;
d→final = (d→final+1)%MAX;
```

A principal diferença entre as inserções Dinâmica e Estática, é que na estática, a função verificará se o tamanho máximo foi atingido, caso tenha atingido, não poderá inserir, senão, irá inserir o elemento na última posição normalmente.

### 2.4. Função Remove Início

```
void remove_inicio(deque *d) {
if (!d || !d→inicio)
{
        printf("Não foi possível remover");
        setbuf(stdin, NULL);
        getchar();
        return;
}
no *novono = d→inicio;
d→inicio = d→inicio→prox;
if (!d→inicio)
        d→final = NULL;
else
        d→inicio→ant = NULL;
free(novono);
d→qtd- - ;
return ;
}
```

A função Remove Início, removerá o primeiro elemento do deque. Primeiramente, ela irá verificar se o deque existe ou se está vazio. Caso esteja vazio ou não exista, imprimirá uma mensagem ao usuário dizendo que não pode remover. Senão, criará um nó auxiliar para armazenar o início. O início assumirá o seu próximo, ou seja, o segundo elemento. Caso ele assuma um valor NULL, o final também assumirá, pois o deque estará vazio, senão, o anterior do início assumirá NULL e por fim, dará free ao nó auxiliar, para liberar a memória do antigo início.

```
        d→inicio = (d→inicio=1)%MAX;
```

A única diferença na implementação Estática, é o fato de que o início irá assumir a próxima posição ao invés de liberar memória como no Dinâmico.

### 2.5. Função Remover Final

```
void remove_final(deque *d) {
if (!d || !d→inicio){
        printf(”Não foi possível remover”);
        setbuf(stdin, NULL);
        getchar();
        return;
}
no *novono = d→final;
if (novono == d→inicio)
        d→inicio = NULL;
        d→final = NULL;
}
else{
        novono→ant→prox = NULL;
        d→final = novono→ant;
}
free(novono);
d→qtd- -;
return ;
}
```

A função Remove Final, removerá o último elemento do deque. Primeiramente, ele verificará se o deque existe ou se ele está vazio. Caso esteja vazio ou não exista, imprimirá uma mensagem ao usuário dizendo que não pode remover. Senão, criará um nó auxiliar para armazenar o final. Se o final for igual o início, o início e o final assumirão NULL, ou seja, o deque ficará vazio e só existia um elemento. Senão, o antigo final assumirá NULL e o final será o elemento anterior. Por fim, dará free no nó auxiliar e liberará a memória do antigo final e decrementará um na quantidade.

```
        d→final- -;
        if (d→final ≺ 0)
        d→final = MAX-1;
```

A única diferença na implementação Estática é o fato de que ele irá recuar uma posição e assumir a anterior, e caso fique negativo, ele assumirá a posição MAX-1, para manter o caráter circular.

### 2.6. Função Imprime Deque

```
void imprime_deque (deque *d)
if (!d→inicio)
        printf(”O deque está vazio”);
        return ;
}
```

```
no *aux = d→inicio;
printf(”Elementos do deque: ”);
while (aux→prox != NULL) {
        printf(”%d”, aux→dados.num);
        aux = aux→prox;
        remove_inicio(d);
}
printf(”%d”, aux→dados.num);
remove_inicio(d);
return ;
}
```

A função Imprime Deque, irá imprimir todos elementos presentes no deque, e por característica da fila, irá remover os elementos também. Primeiramente, irá verificar se o deque está vazio, caso esteja, irá avisar ao usuário que não foi possível imprimir. Senão, irá criar um nó auxiliar com a posição inicial e ir percorrendo a lista, assumindo o próximo valor, e após imprimir cada valor, ele é removido do deque.

```
while (d→dados[d→inicio].num != NULL) {
        printf (”%d”,d→dados[d→inicio].num);
        d→inicio = (d→inicio + 1) % MAX;
        d→qtd- -;
        if (d→qtd == 0)
        d→dados[d→inicio].num = NULL;
}
```

A principal diferença na implementação Estática é o fato de que o início assumirá a próxima posição, ao invés de ser removido do deque, mas o funcionamento é o mesmo.

## 3. Resultados das Aplicações

Apesar das diversas aplicações da estrutura deque, nesse trabalho será exemplificado duas delas. Sendo elas:

### 3.1. Fila de Prioridade

A solução que se baseia em usar um deque para uma fila de prioridade é que não há a necessidade de utilização de duas filas, mas sim de apenas um deque. Na aplicação simples utilizada neste trabalho, o programa recebe dois arquivos contendo dois dados de computadores diferentes. Esses dados são a matrícula de um aluno, o nome e o status (quanto menor o status, maior a prioridade de impressão), ao receber os dados do arquivo, eles são inseridos no deque. O primeiro arquivo é inserido no inicio e o segundo no final do deque, após terminar a leitura e o inserimento dos dados, o programa compara o ”status”do dado contido no inicio com o dado contido no final, e então imprime na tela o

de menor status(maior prioridade), simulando uma impressora que recebe dados de dois computadores diferentes.

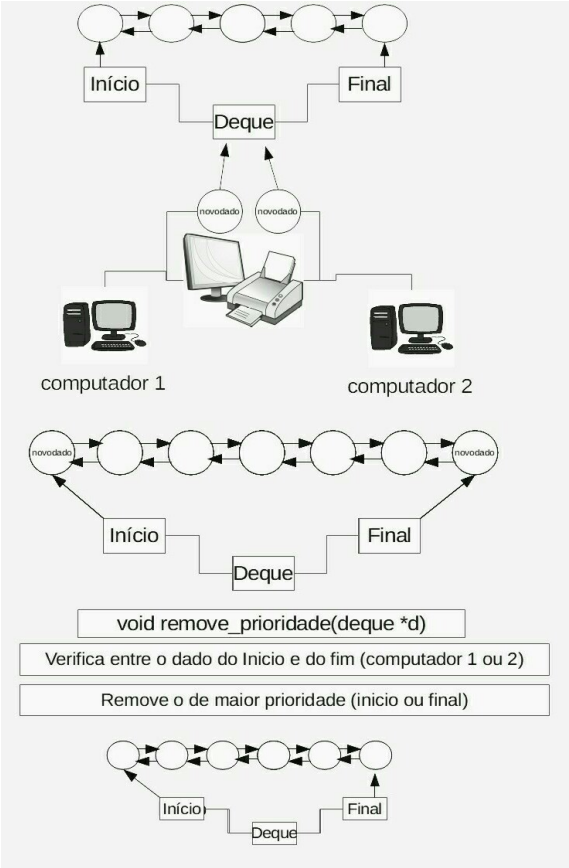


**Figura 1. Exemplificação do problema da fila de prioridade**

### 3.2. Palíndromo

Palíndromos, são palavras, frases ou qualquer outra sequência de unidades, que podem ser lidas da esquerda para a direita ou da direita para a esquerda. Esse exemplo é um clássico quando se trata da estrutura deque. A implementação da mesma é simples, usa-se o deque para o armazenamento de uma string ou um número, e cada caractere como dado de um nó. Após a criação de um segundo deque, faz-se uma cópia do primeiro, retirando o dado pelo final e alocando no inicio do segundo deque, assim terá dois deques, um sendo o inverso do outro. Após a cópia, ocorre a comparação de ambos, podendo ser feita tanto pelo inicio, quanto pelo fim. Caso a comparação detecte que os deques são iguais, o dado fornecido é um palíndromo, caso contrário, não é um palíndromo.

## 4. Conclusão

No decorrer do trabalho foi demonstrado como a estrutura de dados deque se comporta. Foi explicitado algumas aplicações que o mesmo pode ser usado, sendo uma estrutura relativamente simples com suas limitações de inserção e remoção, porém extremamente útil para determinados casos, podendo poupar o esforço computacional. De acordo com a necessidade da aplicação, a inserção e remoção pelos dois extremos trazem uma maior eficiência em relação ao uso de outras estruturas.

## Referências

Backers, A. R. Aula 115 - deque.
https://programacaodescomplicada.wordpress.com/2016/04/04/ed-aula-115-deque-definicao/, Abril.

Brad Miller, D. R. What is a deque?
http://interactivepython.org/runestone/static/pythonds/BasicDS/WhatIsaDeque.html.

Broadhurst, M. Deque in c.
http://www.martinbroadhurst.com/deque.html, Julho.

R. Chandramohan, e. a. Deque and its applications.
https://pt.slideshare.net/jsaddam709/deque-and-its-applications, Novembro.

[Backers ] [Broadhurst ] [R. Chandramohan ] [Brad Miller ]