

Estrutura de Dados Filas

Prof. Thiago Caproni Tavares ¹ Paulo Muniz de Ávila ²

 1 thiago.tavares@ifsuldeminas.edu.br

²paulo.avila@ifsuldeminas.edu.br

Última Atualização: 2 de fevereiro de 2016

Estrutura de Dados

Conteúdo

1 Introdução

- 2 Operações
 - Inicialização
 - Inserindo na Lista
 - Remoções
- 3 Análise da Complexidade

Conteúdo

1 Introdução

- 2 Operações
 - Inicialização
 - Inserindo na Lista
 - Remoções
- 3 Análise da Complexidade

Introdução

- Utiliza uma política FIFO (*first in first out*): o primeiro elemento inserido será o primeiro a ser removido.
- Cada elemento armazena um ou vários dados (estrutura homogênea ou heterogênea) e um ponteiro para o próximo elemento, permitindo o encadeamento e mantendo a estrutura linear;
- Operações: inserir, consultar, remover e esvaziar;
- Possui um ponteiro INÍCIO (remoções) e um FIM (inserções);
 - As operações ocorrem nas duas extremidades da estrutura.

Conteúdo

- 1 Introdução
- 2 Operações
 - Inicialização
 - Inserindo na Lista
 - Remoções
- 3 Análise da Complexidade

Conteúdo

- 1 Introdução
- 2 Operações
 - Inicialização
 - Inserindo na Lista
 - Remoções
- 3 Análise da Complexidade

1º Operação: Inicializa a lista

```
| Fila |
```

1º Operação: Inicializa a lista

```
void inicializar (Fila *fila){
   fila->inicio = NULL;
   fila->fim = NULL;
   fila->tam = 0;
                                                                                                                            lixo
                                                                                                                     inicio fim
```



1º Operação: Inicializa a lista

Conteúdo

- 1 Introdução
- 2 Operações
 - Inicialização
 - Inserindo na Lista
 - Remoções
- 3 Análise da Complexidade

```
int inserir(Fila *fila, int dado){
    cel *novo = malloc(sizeof(cel));

    if(novo = NULL)
        return 0;

    novo->dado = dado;
    novo->prox = NULL;

    if(fila->inicio = NULL){
        fila->finicio = novo;
        fila->finie = novo;
    }

    fla->finie = novo;
    }

    fila->finie = novo;
}

fila->finie = novo;
}

fila->finie = novo;
}
```

```
Fila 500 dado prox lixo lixo NULL NULL 500
```

```
int inserir(Fila *fila, int dado){
    cet *novo = malloc(sizeof(cet));

if(novo = NULL)
    return 0;

novo->dado = dado;
novo->prox = NULL;

if(fila->inicio = NULL){
    fila->inicio = novo;
    fila->inicio = novo;
    fila->fim = novo;
} else{
        fila->fim->prox = novo;
        fila->fim = novo;
}

fila->tam++;
return 1;
}
```

```
Fila 500 | dado | prox | lixo | lixo | lixo | NULL | NULL | 500 |
```

```
int inserir(Fila *fila, int dado){
    cet *novo = malloc(sizeof(cet));
    if(novo = NULL)
        return 0;

    novo->bado = dado;
    novo->prox = NULL){
    if(fila->inicio = NULL){
        fila->inicio = novo;
        fila->fim = novo;
    }
} else{
        fila->fim->prox = novo;
        fila->fim = novo;
}

fila->tam++;
return 1;
}
```

```
int inserir(Fila *fila, int dado){
    cet 'novo = malloc(sizeof(cet));
    if(novo == NULL)
        return 0;
    novo->dado = dado;
    novo->prox = NULL;
    if(fila->inicio == NULL){
        fila->inicio == novo;
        fila->fin == novo;
    }
else{
        fila->fin == novo;
    }
fila->fim = novo;
}
fila->tam++;
return 1;
}
```

```
Fila 500 dado prox 3 NULL NULL NULL NULL 500
```

```
int inserir(Fila *fila, int dado){
    cel *novo = malloc(sizeof(cel));
    if(novo == NULL)
        return 0;
    novo->dado = dado;
    novo->prox = NULL);
    if(fila->inicio == NULL){
        fila->inicio == novo;
        fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fil
```

```
Fila 500 dado prox 3 NULL ► 11 NULL NULL 500
```

```
2º Operação:
             Inserção do #3
                                                 Fila
                                                                 prox
                                                 tam
int inserir(Fila *fila, int dado){
                                                               NULL → II
     cel *novo = malloc(sizeof(cel)):
                                               inicio fim
                                              500 NULL
     if(novo == NULL)
          return θ;
     novo->dado = dado;
     novo->prox = NULL;
     if(fila->inicio == NULL){
         fila->inicio = novo;
          fila->fim = novo;
     }else{
          fila->fim->prox = novo:
          fila->fim = novo:
     fila->tam++:
     return 1;
```

```
2º Operação:
Inserção do #3
```

```
int inserir(Fila *fila, int dado){
    cel *novo = malloc(sizeof(cel));
    if(novo = NULL)
        return 0;
    novo->dado = dado;
    novo->prox = NULL){
        fila->inicio = NULL){
        fila->inicio = novo;
        fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila-
```

```
2º Operação:
             Inserção do #3
                                                 Fila
                                                                prox
                                                 tam
int inserir(Fila *fila, int dado){
                                                               NULL → II
     cel *novo = malloc(sizeof(cel)):
                                               inicio fim
                                              500 500
     if(novo == NULL)
          return θ;
     novo->dado = dado;
     novo->prox = NULL;
     if(fila->inicio == NULL){
          fila->inicio = novo:
          fila->fim = novo;
     }else{
          fila->fim->prox = novo:
          fila->fim = novo:
    fila->tam++;
     return 1;
```

2º Operação: Inserção do #3

```
int inserir(Fila *fila, int dado){
     cel *novo = malloc(sizeof(cel)):
     if(novo == NULL)
          return θ;
     novo->dado = dado;
     novo->prox = NULL;
     if(fila->inicio == NULL){
         fila->inicio = novo:
          fila->fim = novo;
     }else{
          fila->fim->prox = novo:
          fila->fim = novo:
     fila->tam++;
    return 1;
```

```
prox
   tam
                  NULL → II
inicio fim
500 500
```

Fila

```
int inserir(Fila *fila, int dado){
    cel *novo = malloc(sizeof(cel));

    if(novo = NULL)
        return 0;

    novo->dado = dado;
    novo->prox = NULL)
    if(fila->inicio = NULL){
        fila->inicio = novo;
        fila->fim = novo;
    }
}else{
        fila->fim->prox = novo;
        fila->fim = novo;
}

fila->tom++;
return 1;
}
```

```
int inserir(Fila *fila, int dado){
    col *novo = malloc(sizeof(cel));

    if(novo = NULL)
        return 0;

    novo->dado = dado;
    novo->prox = NULL)
    if(fila->inicio = NULL){
        fila->inicio = novo;
        fila->fim = novo;
    }
else{
        fila->fim = novo;
    }
    fila->tam++;
    return 1;
}
```

```
int inserir(Fila *fila, int dado){
    cel *novo = malloc(sizeof(cel));

    if(novo = NULL)
        return 0;

    novo->dado = dado;
    novo->prox = MULL);

    if(fila->inicio = MULL){
        fila->inicio = novo;
        fila->fim = novo;
    }
else{
        fila->fim->prox = novo;
        fila->fim = novo;
}

    fila->tam++;
    return 1;
}
```

```
Fila 500 650 dado prox 3 NULL → II 8 loco 650 650
```

```
int inserir(Fila *fila, int dado){
    cel *novo = malloc(sizeof(cel));

    if(novo = NULL)
        return 0;

    novo->dado = dado;
    novo->prox = NULL);

    if(fila->inicio == NULL){
        fila->inicio = novo;
        fila->fim = novo;
    }
else{
        fila->fim->prox = novo;
        fila->fim = novo;
}

fila->tam++;
return 1;
}
```

```
Fila 500 650 650 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1 650 1
```

3º Operação: Inserção do #8

```
int inserir(Fila *fila, int dado){
     cel *novo = malloc(sizeof(cel)):
     if(novo == NULL)
          return θ;
     novo->dado = dado;
     novo->prox = NULL;
    if(fila->inicio == NULL){
          fila->inicio = novo:
          fila->fim = novo;
     }else{
          fila->fim->prox = novo:
          fila->fim = novo:
     fila->tam++:
     return 1;
```

```
500 500
```

Fila tam

inicio fim

```
3º Operação:
             Inserção do #8
                                                Fila
                                                 tam
int inserir(Fila *fila, int dado){
     cel *novo = malloc(sizeof(cel)):
                                               inicio fim
                                              500 500
     if(novo == NULL)
          return θ;
     novo->dado = dado;
     novo->prox = NULL;
     if(fila->inicio == NULL){
          fila->inicio = novo:
          fila->fim = novo;
     }else{
         fila->fim->prox = novo:
          fila->fim = novo:
     fila->tam++:
     return 1;
```

```
3º Operação:
             Inserção do #8
                                                 Fila
                                                 tam
int inserir(Fila *fila, int dado){
     cel *novo = malloc(sizeof(cel)):
                                               inicio fim
                                              500 650
     if(novo == NULL)
          return θ;
     novo->dado = dado;
     novo->prox = NULL;
     if(fila->inicio == NULL){
          fila->inicio = novo:
          fila->fim = novo;
     }else{
          fila->fim->prox = novo:
         fila->fim = novo;
     fila->tam++:
     return 1;
```

```
3º Operação:
             Inserção do #8
                                                Fila
                                                 tam
int inserir(Fila *fila, int dado){
     cel *novo = malloc(sizeof(cel)):
                                               inicio fim
                                              500 650
     if(novo == NULL)
          return θ;
     novo->dado = dado;
     novo->prox = NULL;
     if(fila->inicio == NULL){
          fila->inicio = novo:
          fila->fim = novo;
     }else{
          fila->fim->prox = novo:
          fila->fim = novo:
    fila->tam++;
    return 1;
```

```
3º Operação:
Inserção do #8
```

```
int inserir(Fila *fila, int dado){
     cel *novo = malloc(sizeof(cel)):
     if(novo == NULL)
          return θ;
     novo->dado = dado;
     novo->prox = NULL;
     if(fila->inicio == NULL){
         fila->inicio = novo:
          fila->fim = novo;
     }else{
          fila->fim->prox = novo:
          fila->fim = novo:
     fila->tam++;
    return 1;
```

```
inicio fim
500 650
```

Fila tam

```
4º Operação:
Inserção do #5
```

```
int inserir(fila *fila, int dado){
    cel *novo = malloc(sizeof(cel));
    if(novo = NULL)
        return 0;
    novo->dado = dado;
    novo->prox = NULL;
    if(fila->inicio = NULL){
        fila->inicio = novo;
        fila->fim = novo;
    }
else{
        fila->fim->prox = novo;
        fila->fim = novo;
    }
    fila->tom++;
    return 1;
}
```

```
int inserir(Fila *fila, int dado){
    cel *novo = malloc(sizeof(cel));
    if(novo == NULL)
        return 0;
    novo->dado = dado;
    novo->prox = NULL;
    if(fila->inicio == NULL){
        fila->inicio == novo;
        fila->fin->prox == novo;
        fila->fin->prox == novo;
        fila->fin == novo;
    }
}
fila->tam++;
return 1;
}
```

```
int inserir(Fila *fila, int dado){
    cel *novo = malloc(sizeof(cel));
    if(novo = NULL)
        return 0;
        novo->dado = dado;
        novo->prox = NULL;
    if(fila->inicio = NULL){
        fila->inicio = novo;
        fila->fin->prox = novo;
        fila->fin = novo;
    }
else{
        fila->fin = novo;
    }
    fila->tam++;
    return 1;
}
```

Fila tam

```
4º Operação:
Inserção do #5
```

```
int inserir(Fila *fila, int dado){
     cel *novo = malloc(sizeof(cel)):
     if(novo == NULL)
          return θ;
     novo->dado = dado;
     novo->prox = NULL;
     if(fila->inicio == NULL){
         fila->inicio = novo:
          fila->fim = novo;
     }else{
          fila->fim->prox = novo:
          fila->fim = novo:
     fila->tam++:
     return 1;
```

```
inicio fim
500 650
```

```
4º Operação:
Inserção do #5
```

```
int inserir(Fila *fila, int dado){
    cel *novo = malloc(sizeof(cel));

    if(novo = NULL)
        return 0;

    novo->dado = dado;
    novo->prox = NULL);

    if(fila->inicio = NULL){
        fila->inicio = novo;
        fila->fina-prox = novo;
        fila->fin = novo;
    }

    fila->tam++;
    return 1;
}
```

```
4º Operação:
Inserção do #5
```

```
int inserir(Fila *fila, int dado){
    cel *novo = malloc(sizeof(cel));
    if(novo = NULL)
        return 0;
    novo->dado = dado;
    novo->prox = NULL;
    if(fila->inicio = NULL){
        fila->inicio = novo;
        fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->fila->f
```

```
4º Operação:
             Inserção do #5
                                                 Fila
                                                 tam
int inserir(Fila *fila, int dado){
     cel *novo = malloc(sizeof(cel)):
                                               inicio fim
                                              500 700
     if(novo == NULL)
          return θ;
     novo->dado = dado;
     novo->prox = NULL;
     if(fila->inicio == NULL){
          fila->inicio = novo:
          fila->fim = novo;
     }else{
          fila->fim->prox = novo:
         fila->fim = novo;
     fila->tam++:
     return 1;
```

Fila tam

inicio fim 500 700

```
4º Operação:
        Inserção do #5
if(novo == NULL)
```

```
int inserir(Fila *fila, int dado){
     cel *novo = malloc(sizeof(cel)):
          return θ;
     novo->dado = dado;
     novo->prox = NULL;
     if(fila->inicio == NULL){
         fila->inicio = novo:
          fila->fim = novo;
     }else{
          fila->fim->prox = novo:
          fila->fim = novo:
    fila->tam++:
     return 1;
```

Fila

inicio fim 500 700

```
500 prox dado prox dado prox 8 700 5 NULL
```

```
5º Operação:
Inserção do #1
```

```
int inserir(Fila *fila, int dado){
    cel *novo = malloc(sizeof(cel));
    if(novo = NULL)
        return 0;
    novo->dado = dado;
    novo->prox = NULL;
    if(fila->inicio = NULL){
        fila->inicio = novo;
        fila->fin = novo;
    }
else{
        fila->fin->prox = novo;
        fila->fin = novo;
    }

    fila->tam++;
    return 1;
}
```

Fila tam

inicio fim

```
int inserir(Fila *fila, int dado){
     cel *novo = malloc(sizeof(cel)):
    if(novo == NULL)
          return θ;
     novo->dado = dado;
     novo->prox = NULL;
     if(fila->inicio == NULL){
         fila->inicio = novo:
          fila->fim = novo;
     }else{
          fila->fim->prox = novo:
          fila->fim = novo:
     fila->tam++:
     return 1;
```

```
500 700
```

Fila tam

inicio fim

```
int inserir(Fila *fila, int dado){
     cel *novo = malloc(sizeof(cel));
     if(novo == NULL)
          return θ;
    novo->dado = dado;
     novo->prox = NULL;
     if(fila->inicio == NULL){
         fila->inicio = novo:
          fila->fim = novo;
     }else{
          fila->fim->prox = novo:
          fila->fim = novo:
     fila->tam++:
     return 1;
```

```
500 700
```

Fila tam

```
int inserir(Fila *fila, int dado){
     cel *novo = malloc(sizeof(cel)):
     if(novo == NULL)
          return θ;
     novo->dado = dado;
    novo->prox = NULL;
     if(fila->inicio == NULL){
         fila->inicio = novo:
          fila->fim = novo;
     }else{
          fila->fim->prox = novo:
          fila->fim = novo:
     fila->tam++:
     return 1;
```

```
inicio fim
500 700
```

Fila tam

inicio fim

```
5º Operação:
Inserção do #1
```

```
int inserir(Fila *fila, int dado){
     cel *novo = malloc(sizeof(cel)):
     if(novo == NULL)
          return θ;
     novo->dado = dado;
     novo->prox = NULL;
    if(fila->inicio == NULL){
          fila->inicio = novo:
          fila->fim = novo;
     }else{
          fila->fim->prox = novo:
          fila->fim = novo:
     fila->tam++:
     return 1;
```

```
500 700
```

Fila

inicio fim 500 700

```
5º Operação:
Inserção do #1
```

```
int inserir(Fila *fila, int dado){
    cel *novo = malloc(sizeof(cel));

    if(novo = NULL)
        return 0;

    novo->dado = dado;
    novo->prox = NULL;

    if(fila->inicio = NULL){
        fila->inicio = novo;
        fila->fila = novo;
    }

else{
        fila->fila = novo;
    }

fila->fila = novo;
}

fila->fila->fila = novo;
}

fila->fila->fila = novo;
}
```

```
500 prox dodo prox dodo prox 5 750 1 NULL 1
```

```
5º Operação:
             Inserção do #1
                                                 Fila
                                                 tam
int inserir(Fila *fila, int dado){
     cel *novo = malloc(sizeof(cel)):
                                               inicio fim
                                              500 750
     if(novo == NULL)
          return θ;
     novo->dado = dado;
     novo->prox = NULL;
     if(fila->inicio == NULL){
          fila->inicio = novo:
          fila->fim = novo;
     }else{
          fila->fim->prox = novo:
         fila->fim = novo;
     fila->tam++:
     return 1;
```

Fila tam

inicio fim 500 750

```
5º Operação:
Inserção do #1
```

```
int inserir(Fila *fila, int dado){
     cel *novo = malloc(sizeof(cel)):
     if(novo == NULL)
          return θ;
     novo->dado = dado;
     novo->prox = NULL;
     if(fila->inicio == NULL){
         fila->inicio = novo:
          fila->fim = novo;
     }else{
          fila->fim->prox = novo:
          fila->fim = novo:
    fila->tam++:
     return 1;
```

Fila

inicio fim 500 750

```
int inserir(Fila *fila, int dado){
    cel *nove = malloc(sizeof(cel));

    if(nove = NULL)
        return 0;

    nove->dado = dado;
    nove->prox = NULL);

    if(fila->inicio == NULL){
        fila->inicio = novo;
        ifila->fin = novo;
        fila->fin = novo;
        fila->fin = novo;
    }

    fila->tamin = novo;
}

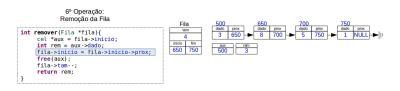
fila->tam++;
return 1;
}
```

```
500 650 700 750 0000 prox 3 650 8 700 5 750 1 NULL
```

Conteúdo

- 1 Introdução
- 2 Operações
 - Inicialização
 - Inserindo na Lista
 - Remoções
- 3 Análise da Complexidade

6° Operação: Remoção da Fila int remover(Fila *fila){ cel *aux = fila->inicio; int reme aux->dado; fila->inicio = fila->inicio->prox; free(aux); fila->tam-; fila->ta



6º Operação: Remoção da Fila int remover(Fila *fila){ cel *aux = fila->inicio: int rem = aux->dado; fila->inicio = fila->inicio->prox: free(aux); fila->tam--;

return rem;

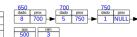
```
inicio fim
650 750
```

Fila tam

6º Operação: Remoção da Fila int remover(Fila *fila){ cel *aux = fila->inicio: int rem = aux->dado; fila->inicio = fila->inicio->prox: free(aux); fila->tam--; return rem;

```
650 750
```

Fila tam



7º Operação: Remoção da Fila int remover(Fila *fîla){ cel.*aux = fîla>sinicio = fila>sinicio = fila>sinic

7º Operação: Remoção da Fila int remover(Fila *fila) { cel *aux = fila>sinicio; int rem = aux->adao; fila>sinicio = fila>sinicio

7º Operação: Remoção da Fila int remover(Fila *fila){ cel *aux = fila->inicio; int rem = aux->dado; fila->inicio = fila->inicio->prox; | Free(aux); | Tree(aux); return rem; }

7º Operação: Remoção da Fila int remover(Fila *fila){ cel *aux = fila->inicio; int rem = aux->dado; fila->inicio = fila->inicio->prox; fre(aux); fila->tam-; return rem;

7º Operação: Remoção da Fila int remover(Fila *fila){ cel *aux = fila->inicio; int rem = aux->dado; fila->inicio = fila->inicio->prox; free(aux); fila->tam:-; return rem; }



8º Operação: Remoção da Fila int remover(Fila *fila){ cel *aux = fila>-inicio; Lint rem = aux-ədado; fila>-inicio = fila>-inicio->prox; free(aux); fila>-tam--; return rem; }

8° Operação: Remoção da Fila int remover(Fila *fila){ cet *aux = fila->inicio; int rem = aux->dado; fila->inicio = fila->inicio->prox; free(aux); fila->tam--; return rem; }

8º Operação: Remoção da Fila int remoção da Fila | 5 | 5 | 5 | cel *aux = fila->inicio; int rem = aux->dado; fila->inicio = fila->inicio->prox; [free(aux); fila->tam-:; return rem; }

8° Operação: Remoção da Fila int remover(Fila *fila){ cel *aux = fila>-inicio; int rem = aux→addo; fila>-inicio = fila>-inicio->prox; free(aux); [fila>-tam=-; return rem; }



Conteúdo

- 1 Introdução
- 2 Operações
 - Inicialização
 - Inserindo na Lista
 - Remoções
- 3 Análise da Complexidade

Análise da Complexidade

- A inserção sempre realiza operações básicas, para atualizar o INÍCIO e FIM da fila;
- O mesmo ocorre no caso da remoção para atualizar o INÍCIO;
 - São operações de tempo constante e gastam O(1).
- Consultar toda a fila percorre os elementos armazenados. Uma fila contém n elementos:
 - logo o tempo de execução é O(n).
- A operação de esvaziamento da fila remove todos os elementos:
 - logo o tempo de execução é O(n).

Obrigado pela atenção!!! thiago.tavares@ifsuldeminas.edu.br



Referências I

- ASCENCIO, A.; CAMPOS, E. de. Fundamentos da programação de computadores: algoritmos, Pascal, C/C++ e Java. Pearson Prentice Hall, 2008. ISBN 9788576051480. Disponível em:
- $<\! \mathsf{https:} // \mathsf{books.google.com.br/books?id} \!\!=\!\! \mathsf{p-mTPgAACAAJ} \!\!>\! .$
- © C: A Reference Manual. Pearson Education, 2007. ISBN 9788131714409. Disponível em: https://books.google.com.br/books? id=Wt2NEypdGNIC>.
- DAMAS, L. *LINGUAGEM C.* LTC. ISBN 9788521615194. Disponível em: https://books.google.com.br/books?id=22-vPgAACAAJ.
- FEOFILOFF, P. Algoritmos Em Linguagem C. CAMPUS
- //books.google.com.br/books?id = LfUQai78VQgC >.
- KERNIGHAN, B.; RITCHIE, D. *C: a linguagem de programação padrão ANSI*. Campus, 1989. ISBN 9788570015860. Disponível em: https://books.google.com.br/books?id=aVWrQwAACAAJ.

Referências II

LOPES, A.; GARCIA, G. *Introdução à programação: 500 algoritmos resolvidos.* Campus, 2002. ISBN 9788535210194. Disponível em: https://books.google.com.br/books?id=Rd-LPgAACAAJ.

//books.google.com.br/books?id=7xt7PgAACAAJ>.

SCHILDT, H.; MAYER, R. *C completo e total*. Makron, 1997. ISBN 9788534605953. Disponível em: https://books.google.com.br/books?id=PbI0AAAACAAJ.