UFSJ - Ciências da Computação

Laboratório de Programação 2

Roteiro 6

Nome: Geraldo Arthur Detomi

1.1) TAD: Deque Sequencial Estático

```
roteiro_6/deque_s_estatico.h
```

```
#ifndef DEQUE H
 2
   #define DEQUE H
 3
 4
   #include <stdio.h>
   #include <stdlib.h>
5
6
 7
   #define MAX 100
8
   typedef struct {
9
     int qtd, ini, fim;
10
11
     int dados[MAX];
12 } Deque;
13
   Deque *criaDeque();
15
16
   void destroiDeque(Deque **dq);
17
18
   int tamanhoDeque(Deque *dq);
19
   int estaCheio(Deque *dq);
20
21
22
   int estaVazio(Deque *dq);
23
   int insereInicio(Deque *dq, int elem);
24
25
26
   int insereFim(Deque *dq, int elem);
27
28
   int removeInicio(Deque *dq);
29
30 int removeFim(Deque *dq);
31
32 int verInicio(Deque *dq, int *p);
33
   int verFim(Deque *dq, int *p);
34
35
36
   void imprime(Deque *dq);
37
38 #endif
roteiro_6/deque_s_estatico.c
```

```
1 #include "./deque s estatico.h"
3 Deque *criaDeque() {
     Deque *dq;
     dq = (Deque *)malloc(sizeof(Deque));
     if (dq != NULL) {
7
        dq \rightarrow qtd = dq \rightarrow ini = dq \rightarrow fim = 0;
8
     }
9
      return dq;
```

```
11
12
   void destroiDeque(Deque **dq) {
13
      if (*dq != NULL) {
14
        free(*dq);
        *dq = NULL;
15
      }
16
17
    }
18
19
   int tamanhoDeque(Deque *dq) {
20
      if (dq == NULL)
21
        return -1;
22
      return dq->qtd;
23
   }
24
25
   int estaCheio(Deque *dq) {
26
      if (dq == NULL)
27
        return -1;
28
      return (dq->qtd == MAX);
   }
29
30
31 int estaVazio(Deque *dq) {
32
      if (dq == NULL)
33
        return -1;
34
      return (dq->qtd == 0);
35
   }
36
37
   int insereInicio(Deque *dq, int elem) {
38
      if (dq == NULL)
39
        return 0;
      if (estaCheio(dq))
40
41
        return 0;
42
      dq->ini = (dq->ini - 1 < 0 ? MAX - 1 : dq->ini - 1);
43
      dq->dados[dq->ini] = elem;
44
      dq->qtd++;
45
      return 1;
46
   }
47
   int insereFim(Deque *dq, int elem) {
48
49
      if (dq == NULL)
50
        return 0;
51
      if (estaCheio(dq))
52
        return 0;
53
      dq->dados[dq->fim] = elem;
54
      dq - sim = (dq - sim + 1) % MAX;
55
      dq->qtd++;
56
      return 1;
57
   }
58
   int removeInicio(Deque *dq) {
59
60
      if (dq == NULL)
61
        return 0;
62
      if (estaVazio(dq))
63
        return 0;
64
      dq \rightarrow ini = (dq \rightarrow ini + 1) % MAX;
      dq->qtd--;
65
66
      return 1;
67
    }
68
69
   int removeFim(Deque *dq) {
70
      if (dq == NULL)
71
        return 0;
72
      if (estaVazio(dq))
73
        return 0;
74
      dq - fim = (dq - fim - 1 < 0 ? MAX - 1 : dq - fim - 1);
75
      dq->qtd--;
```

```
76
       return 1;
 77 | }
 78
 79 int verInicio(Deque *dq, int *p) {
      if (dq == NULL)
 80
81
         return 0;
82
       if (estaVazio(dq))
 83
         return 0;
 84
       *p = dq->dados[dq->ini];
 85
       return 1;
 86
    }
 87
 88
    int verFim(Deque *dq, int *p) {
      if (dq == NULL)
89
 90
         return 0;
 91
       if (estaVazio(dq))
 92
         return 0;
 93
       int i = (dq - fim - 1 < 0 ? MAX - 1 : dq - fim - 1);
       *p = dq->dados[i];
 94
 95
       return 1;
 96 }
97
98
    void imprime(Deque *dq) {
99
       if (dq == NULL)
         return;
100
101
       if (estaVazio(dq)) {
102
         printf("Deque Vazio!\n");
103
         return;
104
       }
105
       int i = dq->ini;
       printf("Elementos: \n");
106
107
108
        printf("%d ", dq->dados[i]);
109
         i = (i + 1) % MAX;
110
       } while (i != dq->fim);
111
       // Usar do..while garante a impressao de todos elementos
112
       // mesmo com a Deque cheia
       printf("\n");
113
114 }
roteiro_6/1-1.c
  1 #include "./deque_s_estatico.h"
  2
  3
    #define MAX OPTIONS 10
  4
  5
    enum options {
  6
      CRIAR FILA = 0,
  7
      INSERIR ITEM,
  8
       INSERIR INICIO,
  9
       SHOW INICIO,
 10
       SHOW FIM,
 11
       REMOVER ITEM,
 12
       REMOVER_INICIO,
 13
       IMPRIMIR,
 14
       DESTRUIR,
 15
       SAIR,
16 };
 17
    int get_option() {
 18
      int opt = -1;
 19
 20
       do {
         printf("\t0peracoes\n");
 21
         printf("[%d] Criar deque, ", CRIAR FILA);
 22
 23
         printf("[%d] Inserir no fim, ", INSERIR ITEM);
         printf("[%d] Inserir no inicio, ", INSERIR_INICIO);
 24
```

printf("Valor inserido com sucesso\n");

88

```
90
91
           break;
92
93
         case INSERIR INICIO:
           printf("Executando comando...\n");
94
95
           if (deque == NULL) {
96
97
             printf("Deque não instanciado impossivel realizar operação..\n");
98
             break;
99
           }
100
           if (!insereInicio(deque, get valor())) {
101
102
             printf("Falha ao inserir no inicio do deque!\n");
103
104
105
           printf("Valor inserido com sucesso!\n");
106
107
          break;
108
         case SHOW INICIO:
109
           printf("Executando comando...\n");
110
           if (deque == NULL) {
111
             printf("Deque não instanciado impossivel realizar operação..\n");
112
113
             break;
114
115
116
           if (!verInicio(deque, &valor)) {
117
             printf("Erro ao ver inicio do deque\n");
118
             break:
119
120
121
           printf("Valor de inicio = %d\n", valor);
122
           break:
         case SHOW FIM:
123
           printf("Executando comando...\n");
124
125
126
           if (deque == NULL) {
             printf("Deque não instanciado impossivel realizar operação..\n");
127
128
             break;
129
130
131
           if (!verFim(deque, &valor)) {
132
             printf("Erro ao ver fim do deque\n");
133
             break;
134
           }
135
136
           printf("Valor do fim do deque = %d\n", valor);
137
138
         case REMOVER ITEM:
139
           printf("Executando comando...\n");
140
141
           if (deque == NULL) {
142
             printf("Deque não instanciado impossivel realizar operação..\n");
143
             break;
144
           }
145
           if (!removeFim(deque)) {
146
147
             printf("Erro ao executar comando\n");
148
149
150
           printf("Removido item do fim do deque com sucesso!\n");
151
152
           break;
153
         case REMOVER INICIO:
154
           printf("Executando comando...\n");
```

```
if (deque == NULL) {
157
             printf("Deque não instanciado impossivel realizar operação..\n");
158
             break;
159
           }
160
           if (!removeInicio(deque)) {
161
162
             printf("Erro ao executar comando\n");
163
164
165
           printf("Removido item do inicio do deque com sucesso!\n");
166
167
           break;
168
         case IMPRIMIR:
           printf("Executando comando...\n");
169
170
171
           if (deque == NULL) {
172
             printf("Deque não instanciado impossivel realizar operação..\n");
173
             break;
174
           }
175
176
           imprime(deque);
177
178
           break;
179
         case DESTRUIR:
180
           printf("Executando comando...\n");
181
182
           destroiDeque(&deque);
183
           deque = NULL;
184
185
           printf("Deque destruido com sucesso!\n");
186
187
           break;
188
         case SAIR:
189
           printf("Finalizando programa! Até mais!\n");
190
           break;
191
         }
192
193
       } while (opt != SAIR);
194
195
       return 0;
196 }
```

Saída do terminal:

```
~/Documents/UFSJ-Graduacao/UFSJ-2025_1/Lab_Prog_2/roteiro_6
                                                                                                                                                                       main/ 25-05-03 - 11:43:17
C.:: ./1-1.out
           Operacoes
     Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Insira a opcão desejada: 0
Executando comando..
Deque criado com sucesso!
          Operacoes
[0] Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Insira a opção desejada: l
Executando comando.
Digite o valor a inserir:l
Valor inserido com sucesso
          Operacoes
[0] Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Insira a opção desejada: 2
Executando comando.
Digite o valor a inserir:2
Valor inserido com sucesso!
          Operacoes
[0] Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Insira a opção desejada: 3
Executando comando..
Valor de inicio = 2
          Operacoes
[0] Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Insira a opção desejada: 6
Executando comando.
Removido item do inicio do deque com sucesso!
         Operacoes 4 1
[0] Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Insira a opção desejada: 7
Executando comando...
Elementos
[0] Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Insira a opção desejada: 8
Executando comando..
Deque destruido com sucesso!
          Operacoes
[0] Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Insira a opção desejada: 1
Executando comando
Deque não instanciado impossivel realizar operação..
          Operacoes
[0] Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Insira a opção desejada:
```

1.2) TAD: Deque Duplamente Encadeado

roteiro_6/deque_encadeado.h

```
#ifndef DEQUE ENCADEADO
1
2
   #define DEQUE ENCADEADO
3
4
   #include <stdio.h>
5
   #include <stdlib.h>
6
7
   typedef struct NO {
8
      int info;
9
      struct NO *prox;
10
      struct NO *ant;
   } NO;
11
12
13
   typedef struct {
14
      int atd;
      struct NO *ini:
15
```

```
struct NO *fim;
16
17
    } Deque;
18
19
   NO *alocarNO();
20
21
    void liberarNO(NO *q);
22
23
    Deque *criaDeque();
24
25
   void destroiDeque(Deque **dq);
26
27
   int tamanhoDeque(Deque *dq);
28
   int estaVazio(Deque *dq);
29
30
31
    int insereInicio(Deque *dq, int elem);
32
33
   int insereFim(Deque *dq, int elem);
34
35
   int removeInicio(Deque *dq);
36
37
    int removeFim(Deque *dq);
38
39
    int verInicio(Deque *dq, int *p);
40
41
    int verFim(Deque *dq, int *p);
42
43
   void imprime(Deque *dq);
44
45 #endif
roteiro_6/deque_encadeado.c
  1 #include "./deque_encadeado.h"
  2
    NO *alocarNO() { return (NO *)malloc(sizeof(NO)); }
  3
  4
  5
    void liberarNO(NO *q) { free(q); }
  6
  7
    Deque *criaDeque() {
  8
       Deque *dq;
  9
       dq = (Deque *)malloc(sizeof(Deque));
       if (dq != NULL) {
 10
 11
         dq \rightarrow qtd = 0;
 12
         dq->ini = NULL;
 13
         dq - > fim = NULL;
 14
       }
 15
       return dq;
    }
 16
17
18
    void destroiDeque(Deque **dq) {
 19
       if (*dq != NULL) {
 20
         NO *aux;
 21
         while ((*dq)->ini != NULL) {
 22
           aux = (*dq) ->ini;
 23
           (*dq)->ini = (*dq)->ini->prox;
 24
           liberarNO(aux);
25
         }
 26
         free(*dq);
 27
         *dq = NULL;
 28
       }
 29
    }
 30
 31
    int tamanhoDeque(Deque *dq) {
       if (dq == NULL)
 32
         return -1;
 33
```

```
5/3/25, 12:00 PM
   34
         return dq->qtd;
   35 }
   36
   37
      int estaVazio(Deque *dq) {
         if (dq == NULL)
  38
   39
           return -1;
         return (dq->qtd == 0);
   40
   41
   42
   43
      int insereInicio(Deque *dq, int elem) {
         if (dq == NULL)
   44
   45
           return 0;
   46
         NO *novo = alocarNO();
   47
         if (novo == NULL)
   48
           return 0;
   49
         novo->info = elem;
   50
         novo->ant = NULL;
   51
         if (estaVazio(dq)) {
   52
          novo->prox = NULL;
   53
           dq->fim = novo;
   54
         } else {
   55
           dq->ini->ant = novo;
           novo->prox = dq->ini;
   56
   57
   58
         dq->ini = novo;
   59
         dq->qtd++;
   60
         return 1;
   61
      }
   62
      int insereFim(Deque *dq, int elem) {
  63
         if (dq == NULL)
   64
   65
           return 0;
   66
         NO *novo = alocarNO();
   67
         if (novo == NULL)
   68
           return 0;
   69
         novo->info = elem;
   70
         novo->prox = NULL;
   71
         if (estaVazio(dq)) {
   72
           novo->ant = NULL;
   73
           dq->ini = novo;
   74
         } else {
   75
           dq->fim->prox = novo;
   76
           novo->ant = dq->fim;
   77
   78
         dq->fim = novo;
   79
         dq->qtd++;
   80
         return 1;
   81
   82
  83
      int removeInicio(Deque *dq) {
   84
         if (dq == NULL)
   85
           return 0;
         if (estaVazio(dq))
   86
   87
           return 0;
         NO *aux = dq->ini;
   88
   89
         if (dq->ini == dq->fim) {
   90
           dq->ini = dq->fim = NULL;
   91
         } else {
   92
           dq->ini = dq->ini->prox;
   93
           dq->ini->ant = NULL;
   94
   95
         liberarNO(aux);
   96
         dq->qtd--;
   97
         return 1;
   98 }
```

```
5/3/25, 12:00 PM
   99
  100
      int removeFim(Deque *dq) {
  101
         if (dq == NULL)
  102
           return 0;
  103
         if (estaVazio(dq))
  104
           return 0;
         NO *aux = dq->fim;
  105
  106
         if (dq->ini == dq->fim) {
  107
           dq->ini = dq->fim = NULL;
  108
         } else {
  109
           dq->fim = dq->fim->ant;
  110
           dq->ini->prox = NULL;
  111
         liberarNO(aux);
 112
  113
         dq->qtd--;
  114
         return 1;
 115
      }
 116
 117
      int verInicio(Deque *dq, int *p) {
 118
         if (dq == NULL)
 119
           return 0:
 120
         if (estaVazio(dq))
  121
           return 0;
  122
         *p = dq->ini->info;
 123
         return 1;
 124 }
 125
  126
      int verFim(Deque *dq, int *p) {
        if (dq == NULL)
 127
  128
           return 0;
  129
         if (estaVazio(dq))
  130
           return 0;
 131
         *p = dq->fim->info;
         return 1;
 132
 133 }
 134
 135
      void imprime(Deque *dq) {
        if (dq == NULL)
 136
 137
           return;
  138
         if (estaVazio(dq)) {
 139
           printf("Deque Vazio!\n");
  140
           return;
  141
  142
         N0 *aux = dq->ini;
  143
         printf("Elementos:\n");
        while (aux != NULL) {
  144
          printf("%d ", aux->info);
  145
  146
           aux = aux->prox;
  147
  148
         printf("\n");
  149 }
  roteiro_6/1-2.c
   1 #include "./deque encadeado.h"
   2
   3
      #define MAX OPTIONS 10
   4
   5
      enum options {
   6
         CRIAR FILA = 0,
   7
         INSERIR ITEM,
   8
         INSERIR INICIO,
   9
         SHOW INICIO,
   10
         SHOW FIM,
         REMOVER ITEM,
   11
   12
         REMOVER_INICIO,
```

```
13
      IMPRIMIR,
14
      DESTRUIR,
15
      SAIR,
16 };
17
   int get option() {
18
      int opt = -1;
19
20
      do {
21
        printf("\t0peracoes\n");
22
        printf("[%d] Criar deque, ", CRIAR_FILA);
23
        printf("[%d] Inserir no fim, ", INSERIR_ITEM);
24
        printf("[%d] Inserir no inicio, ", INSERIR INICIO);
        printf("[%d] Ver o inicio do deque, ", SHOW_INICIO);
25
26
        printf("[%d] Ver o fim do deque, ", SHOW_FIM);
27
        printf("\n[%d] Remover do fim do deque, ", REMOVER_ITEM);
        printf("[%d] Remover do inicio do deque, ", REMOVER_INICIO);
28
29
        printf("[%d] Imprimir deque, ", IMPRIMIR);
30
        printf("[%d] Destruir deque, ", DESTRUIR);
31
        printf("[%d] Sair do programa \n", SAIR);
32
33
        printf("\nInsira a opção desejada: ");
34
        scanf("%d", &opt);
35
        printf("\n");
36
37
        if (opt < 0 || opt >= MAX OPTIONS) {
          printf("Opção escolhida inválida!\n");
38
39
40
      } while (opt < 0 || opt >= MAX OPTIONS);
41
42
43
      return opt;
44
45
   int get_valor() {
46
47
      int value;
48
49
      printf("Digite o valor a inserir:");
      scanf("%d", &value);
50
51
52
      return value;
53
   }
54
55
   int main() {
56
     int opt, valor;
57
      Deque *deque = NULL;
58
59
60
      do {
61
        opt = get option();
62
        switch (opt) {
63
64
        case CRIAR FILA:
          printf("Executando comando...\n");
65
66
          if (deque == NULL) {
67
68
            deque = criaDeque();
69
          } else {
70
            destroiDeque(&deque);
71
            deque = NULL;
72
            deque = criaDeque();
73
74
75
          printf("Deque criado com sucesso!\n");
76
          break;
77
        case INSERIR ITEM:
```

```
78
           printf("Executando comando...\n");
79
80
           if (deque == NULL) {
81
             printf("Deque não instanciado impossivel realizar operação..\n");
82
             break;
           }
83
84
85
           if (!insereFim(deque, get valor())) {
             printf("Falha ao inserir no fim do deque!\n");
86
87
88
           printf("Valor inserido com sucesso\n");
89
90
91
           break;
92
93
         case INSERIR INICIO:
94
           printf("Executando comando...\n");
95
96
           if (deque == NULL) {
97
             printf("Deque não instanciado impossivel realizar operação..\n");
98
             break;
99
           }
100
101
           if (!insereInicio(deque, get_valor())) {
             printf("Falha ao inserir no inicio do deque!\n");
102
103
104
105
           printf("Valor inserido com sucesso!\n");
106
107
          break;
         case SHOW INICIO:
108
109
           printf("Executando comando...\n");
110
111
           if (deque == NULL) {
             printf("Deque não instanciado impossivel realizar operação...\n");
112
113
             break;
114
115
           if (!verInicio(deque, &valor)) {
116
117
             printf("Erro ao ver inicio do deque\n");
118
             break:
119
           }
120
121
           printf("Valor de inicio = %d\n", valor);
122
          break;
         case SHOW FIM:
123
124
           printf("Executando comando...\n");
125
126
           if (deque == NULL) {
127
             printf("Deque não instanciado impossivel realizar operação..\n");
128
             break;
129
130
131
           if (!verFim(deque, &valor)) {
132
             printf("Erro ao ver fim do deque\n");
133
             break;
134
135
136
           printf("Valor do fim do deque = %d\n", valor);
137
          break;
138
         case REMOVER ITEM:
139
           printf("Executando comando...\n");
140
141
           if (deque == NULL) {
142
             printf("Deque não instanciado impossivel realizar operação..\n");
```

Saída do terminal:

return 0;

} while (opt != SAIR);

192

193 194 195

196 }

```
L.:: ./1-2.out
           Operacoes
operaces
[0] Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Insira a opção desejada: θ
Executando comando.
Deque criado com sucesso!
          Operacoes
[0] Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Executando comando...
Digite o valor a inserir:2
Valor inserido com sucesso
          Operacoes
operaces
[0] Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Executando comando.
Digite o valor a inserir:1
Valor inserido com sucesso
           Operacoes
[0] Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Insira a opção desejada: 7
Executando comando...
Elementos:
[0] Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Insira a opção desejada: 3
Executando comando
Valor de inicio = 1
           Operacoes
operaces
[0] Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Insira a opção desejada: 5
Executando comando...
Removido item do fim do deque com sucesso!
[0] Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Insira a opcão desejada: 6
Executando comando
Removido item do inicio do deque com sucesso!
           Operacoes
[0] Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Insira a opção desejada: 7
Executando comando...
Deque Vazio!
          Operacoes
[0] Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Insira a opção desejada: 8
Deque destruido com sucesso!
          Operacoes
[0] Criar deque, [1] Inserir no fim, [2] Inserir no inicio, [3] Ver o inicio do deque, [4] Ver o fim do deque,
[5] Remover do fim do deque, [6] Remover do inicio do deque, [7] Imprimir deque, [8] Destruir deque, [9] Sair do programa
Insira a opção desejada: 9
```

2.1) TAD: Fila de Prioridades Simplesmente Encadeada

roteiro_6/fila_prioridade.h

```
#ifndef FILA PRIORIDADE
1
2
   #define FILA PRIORIDADE
3
4
  #include <stdio.h>
5
  #include <stdlib.h>
6
7
  typedef struct NO {
8
    int info, prio;
9
     struct NO *prox;
```

```
11
12
   typedef struct NO *FilaP;
13
14 NO *alocarNO();
15
   void liberarNO(NO *q);
16
17
18 FilaP *criaFila();
19
20 int estaVazia(FilaP *fp);
21
22 int inserirPrio(FilaP *fp, int elem, int pri);
23
24
   int removeInicio(FilaP *fp);
25
26 int verInicio(FilaP *fp, int *p, int *pri);
27
28 void imprime(FilaP *fp);
29
30 void destroiFila(FilaP **fp);
31
32 int tamanhoFila(FilaP *fp);
33
34 #endif
roteiro_6/fila_prioridade.c
  1 #include "./fila prioridade.h"
  2
  3
    NO *alocarNO() { return (NO *)malloc(sizeof(NO)); }
    void liberarNO(NO *q) { free(q); }
  5
  6
  7
    FilaP *criaFila() {
  8
      FilaP *fp;
  9
      fp = (FilaP *)malloc(sizeof(FilaP));
 10
      if (fp != NULL)
        *fp = NULL;
 11
 12
       return fp;
13 }
14
 15
    int estaVazia(FilaP *fp) {
 16
      if (fp == NULL)
17
        return -1;
18
      return ((*fp) == NULL);
19
    }
20
    int inserirPrio(FilaP *fp, int elem, int pri) {
21
22
      if (fp == NULL)
23
        return 0;
 24
      NO *novo = alocarNO();
25
      if (novo == NULL)
 26
        return 0;
 27
 28
      novo->info = elem;
 29
      novo->prio = pri;
30
      if (estaVazia(fp)) {
 31
 32
        novo->prox = *fp;
        *fp = novo;
 33
 34
      } else {
35
        NO *aux, *ant;
 36
        ant = NULL;
 37
        aux = *fp; // Inicio
 38
        while (aux != NULL && aux->prio >= novo->prio) {
```

10 } NO;

```
39
           ant = aux;
40
           aux = aux -> prox;
41
42
         if (ant == NULL) {
43
           novo->prox = *fp;
44
           *fp = novo;
45
         } else {
46
           novo->prox = ant->prox;
47
           ant->prox = novo;
48
49
       }
50
       return 1;
51
    }
52
    int removeInicio(FilaP *fp) {
53
54
       if (fp == NULL)
55
         return 0;
56
       if (estaVazia(fp))
57
         return 0;
58
       N0 *aux = *fp;
59
       *fp = aux->prox;
60
       liberarNO(aux);
       return 1;
61
62
63
64
    int verInicio(FilaP *fp, int *p, int *pri) {
65
       if (fp == NULL)
66
         return 0;
       if (estaVazia(fp))
67
68
         return 0;
       *p = (*fp) -> info;
69
70
       *pri = (*fp)->prio;
71
       return 1;
72
    }
73
    void imprime(FilaP *fp) {
74
75
       if (fp == NULL)
76
         return;
77
       if (estaVazia(fp)) {
78
         printf("Fila Vazia!\n");
79
         return;
80
       }
81
       N0 *aux = *fp;
82
       while (aux != NULL) {
83
         printf("[%d, %d] ", aux->prio, aux->info);
84
         aux = aux->prox;
85
86
       printf("\n");
87
    }
88
    void destroiFila(FilaP **fp) {
89
90
       if (*fp != NULL) {
91
         NO *aux;
92
         while ((**fp) != NULL) {
93
           aux = **fp;
94
           **fp = (**fp) -> prox;
95
           liberarNO(aux);
96
         }
97
         free(*fp);
98
         *fp = NULL;
99
       }
    }
100
101
102
    int tamanhoFila(FilaP *fp) {
       if (fp == NULL)
```

```
104
         return -1;
105
106
       int t = 0;
107
       NO *i = *fp;
       while (i != NULL) {
108
109
        t++;
         i = i -> prox;
110
111
112
113
       return t;
114 }
roteiro_6/2-1.c
  1 #include "fila prioridade.h"
    #define MAX OPTIONS 8
  4
  5
    enum options {
  6
      CRIAR FILA = 0,
  7
       INSERIR ITEM,
  8
       SHOW INICIO,
  9
       REMOVER_ITEM,
 10
       IMPRIMIR,
       SHOW TAMANHO,
 11
 12
       DESTRUIR,
 13
       SAIR,
 14
    };
 15
 16
    int get_option() {
 17
      int opt = -1;
 18
 19
         printf("\t0peracoes\n");
20
         printf("[%d] Criar fila, ", CRIAR_FILA);
21
         printf("[%d] Inserir um item pela prioridade, ", INSERIR_ITEM);
 22
         printf("[%d] Ver o início da Fila, ", SHOW_INICIO);
 23
         printf("[%d] Remover um item, \n", REMOVER_ITEM);
 24
         printf("[%d] Imprimir a Fila, ", IMPRIMIR);
 25
         printf("[%d] Mostrar o tamanho da Fila, ", SHOW TAMANHO);
 26
         printf("[%d] Destruir a Fila, ", DESTRUIR);
 27
         printf("[%d] Sair do programa \n", SAIR);
 28
 29
         printf("\nInsira a opção desejada: ");
 30
         scanf("%d", &opt);
 31
         printf("\n");
 32
         if (opt < 0 || opt >= MAX_OPTIONS) {
 33
 34
           printf("Opção escolhida inválida!\n");
 35
 36
37
       } while (opt < 0 || opt >= MAX OPTIONS);
 38
 39
       return opt;
 40
    }
 41
 42
    int get_valor(char *msg) {
 43
      int value;
 44
       printf("%s\n", msg);
 45
       scanf("%d", &value);
 46
 47
 48
       return value;
 49
    }
 50
    int main() {
 51
       int opt, valor, priori;
```

```
53
54
       FilaP *fila priori = NULL;
55
56
       do {
57
         opt = get_option();
58
59
         switch (opt) {
60
         case CRIAR FILA:
61
           printf("Executando comando...\n");
62
63
           if (fila priori == NULL) {
64
             fila priori = criaFila();
65
           } else {
66
             destroiFila(&fila_priori);
67
             fila_priori = NULL;
68
             fila priori = criaFila();
69
70
           printf("Fila de prioridade criada com sucesso!\n");
71
72
73
         case INSERIR ITEM:
74
           printf("Executando comando...\n");
75
76
           if (fila priori == NULL) {
77
             printf("Fila de prioridade não instanciado impossivel realizar "
78
                     "operação..\n");
79
             break;
80
           }
81
           if (!inserirPrio(fila priori, get valor("Valor a inserir:"),
82
                             get valor("Insira a prioridade: "))) {
83
84
             printf("Falha ao inserir na Fila de prioridade!\n");
85
86
           printf("Item inserido com sucesso\n");
87
88
89
           break;
90
91
         case SHOW_INICIO:
92
           printf("Executando comando...\n");
93
94
           if (fila priori == NULL) {
95
             printf("Fila de prioridade não instanciado impossivel realizar "
96
                     "operação..\n");
97
             break;
98
           }
99
100
           if (!verInicio(fila_priori, &valor, &priori)) {
             printf("Erro ao ver inicio do Fila de prioridade\n");
101
102
             break;
103
           }
104
           printf("Item [valor = %d, prioridade = %d]\n", valor, priori);
105
106
           break;
107
108
         case REMOVER ITEM:
109
           printf("Executando comando...\n");
110
           if (fila priori == NULL) {
111
112
             printf("Fila de prioridade não instanciado impossivel realizar "
113
                     "operação..\n");
114
             break;
115
           }
116
117
           if (!removeInicio(fila priori)) {
```

Saída do terminal:

```
[0] Criar fila, [1] Inserir um item pela prioridade, [2] Ver o início da Fila, [3] Remover um item,
[4] Imprimir a Fila, [5] Mostrar o tamanho da Fila, [6] Destruir a Fila, [7] Sair do programa
Executando comando..
Fila de prioridade criada com sucesso!
Operacoes
[0] Criar fila, [1] Inserir um item pela prioridade, [2] Ver o início da Fila, [3] Remover um item,
[4] Imprimir a Fila, [5] Mostrar o tamanho da Fila, [6] Destruir a Fila, [7] Sair do programa
Insira a opção desejada: l
Insira a prioridade:
Valor a inserir:
Item inserido com sucesso
Operacoes
[0] Criar fila, [1] Inserir um item pela prioridade, [2] Ver o início da Fila, [3] Remover um item,
[4] Imprimir a Fila, [5] Mostrar o tamanho da Fila, [6] Destruir a Fila, [7] Sair do programa
Insira a opção desejada: l
Executando comando..
Insira a prioridade:
100
Valor a inserir:
Item inserido com sucesso
Operacoes
[0] Criar fila, [1] Inserir um item pela prioridade, [2] Ver o início da Fila, [3] Remover um item,
[4] Imprimir a Fila, [5] Mostrar o tamanho da Fila, [6] Destruir a Fila, [7] Sair do programa
Executando comando..
[100, 0] [10, 1]
Operacoes
[0] Criar fila, [1] Inserir um item pela prioridade, [2] Ver o início da Fila, [3] Remover um item,
[4] Imprimir a Fila, [5] Mostrar o tamanho da Fila, [6] Destruir a Fila, [7] Sair do programa
Insira a opção desejada: 5
Executando comando.
Tamanho da fila = 2
           Operacoes
operacoes
[0] Criar fila, [1] Inserir um item pela prioridade, [2] Ver o início da Fila, [3] Remover um item,
[4] Imprimir a Fila, [5] Mostrar o tamanho da Fila, [6] Destruir a Fila, [7] Sair do programa
Insira a opção desejada: 2
Executando comando.
Item [valor = \theta, prioridade = 100]
Operacoes
[0] Criar fila, [1] Inserir um item pela prioridade, [2] Ver o início da Fila, [3] Remover um item,
[4] Imprimir a Fila, [5] Mostrar o tamanho da Fila, [6] Destruir a Fila, [7] Sair do programa
Insira a opção desejada: 3
Executando comando...
Removido primeiro item da Fila de prioridade com sucesso!
[0] Criar fila, [1] Inserir um item pela prioridade, [2] Ver o início da Fila, [3] Remover um item,
[4] Imprimir a Fila, [5] Mostrar o tamanho da Fila, [6] Destruir a Fila, [7] Sair do programa
Insira a opcão desejada: 5
Executando comando.
Tamanho da fila = 1
            Operacoes
[0] Criar fila, [1] Inserir um item pela prioridade, [2] Ver o início da Fila, [3] Remover um item,
[4] Imprimir a Fila, [5] Mostrar o tamanho da Fila, [6] Destruir a Fila, [7] Sair do programa
Insira a opção desejada: 6
Executando comando..
Fila de prioridade destruida com sucesso!
Operacoes
```

2.2) TAD: Fila de Prioridades com Heap Binária

roteiro_6/fila_priori_heap.h

```
#ifndef FILA PRIORI HEAP
1
2
   #define FILA PRIORI HEAP
3
4
   #include <stdio.h>
5
   #include <stdlib.h>
6
7
   #define MAX 100
8
9
   typedef struct NO {
10
     int info, prio;
```

```
11 } NO;
12
13 typedef struct {
14
     int qtd;
    NO dados[MAX];
15
16 } FilaP;
17
18 FilaP *criaFila();
19
20 void destroiFila(FilaP **fp);
21
22 int tamanhoFila(FilaP *fp);
23
   int estaCheia(FilaP *fp);
24
25
26
   int estaVazia(FilaP *fp);
27
28 void trocaNO(NO *a, NO *b);
29
30
   void ajustaHeapInsere(FilaP *fp, int filho);
31
32
   int inserirPrio(FilaP *fp, int elem, int pri);
33
34
   void ajustaHeapRemove(FilaP *fp, int pai);
35
36
   int removeInicio(FilaP *fp);
37
38 int verInicio(FilaP *fp, int *valor, int *pri);
39
40 void imprime(FilaP *fp);
41
42 #endif
roteiro_6/fila_priori_heap.c
  1 #include "./fila_priori_heap.h"
  2
  3
    FilaP *criaFila() {
  4
      FilaP *fp;
  5
      fp = (FilaP *)malloc(sizeof(FilaP));
       if (fp != NULL)
  6
  7
         fp \rightarrow qtd = 0;
  8
       return fp;
    }
  9
 10
 11 void destroiFila(FilaP **fp) {
 12
      if (*fp != NULL) {
 13
         free(*fp);
         *fp = NULL;
14
15
       }
16 }
 17
    int tamanhoFila(FilaP *fp) {
 18
19
       if (fp == NULL)
20
         return -1;
 21
       return fp->qtd;
 22
    }
23
24 int estaCheia(FilaP *fp) {
25
      if (fp == NULL)
 26
         return -1;
       return (fp->qtd == MAX);
 27
28 }
 29
 30 int estaVazia(FilaP *fp) {
 31
       if (fp == NULL)
```

```
32
        return -1;
33
      return (fp->qtd == 0);
34
   }
35
36 void trocaNO(NO *a, NO *b) {
37
      NO temp;
      temp.info = a->info;
38
39
      temp.prio = a->prio;
40
      a->info = b->info;
      a->prio = b->prio;
41
42
      b->info = temp.info;
43
      b->prio = temp.prio;
44
   }
45
46
   void ajustaHeapInsere(FilaP *fp, int filho) {
47
      int pai = (filho - 1) / 2;
48
      int prioPai = fp->dados[pai].prio;
      int prioFilho = fp->dados[filho].prio;
49
50
      while (filho > 0 && prioPai < prioFilho) {
51
        trocaNO(&fp->dados[filho], &fp->dados[pai]);
52
        filho = pai;
53
        pai = (pai - 1) / 2;
54
        prioPai = fp->dados[pai].prio;
        prioFilho = fp->dados[filho].prio;
55
56
      }
57
   }
58
59
   int inserirPrio(FilaP *fp, int elem, int pri) {
60
      if (fp == NULL)
61
        return 0;
      if (estaCheia(fp))
62
63
        return 0;
64
      fp->dados[fp->qtd].info = elem;
      fp->dados[fp->qtd].prio = pri;
65
      ajustaHeapInsere(fp, fp->qtd);
66
67
      fp->qtd++;
68
      return 1;
   }
69
70
71
   void ajustaHeapRemove(FilaP *fp, int pai) {
72
      int filho = 2 * pai + 1;
73
      while (filho < fp->qtd) {
74
        if (filho < fp->qtd - 1)
75
          if (fp->dados[filho].prio < fp->dados[filho + 1].prio)
76
            filho++;
77
78
        if (fp->dados[pai].prio > fp->dados[filho].prio)
79
80
81
        trocaNO(&fp->dados[pai], &fp->dados[filho]);
82
        pai = filho;
83
        filho = 2 * pai + 1;
      }
84
   }
85
86
87
   int removeInicio(FilaP *fp) {
88
      if (fp == NULL)
89
        return 0;
90
      if (estaVazia(fp))
91
        return 0;
92
93
      fp->qtd--;
94
      fp->dados[0].info = fp->dados[fp->qtd].info;
95
      fp->dados[0].prio = fp->dados[fp->qtd].prio;
      ajustaHeapRemove(fp, 0);
```

```
97
      return 1;
98 }
99
100 int verInicio(FilaP *fp, int *valor, int *pri) {
101
      if (fp == NULL)
102
         return 0;
      if (estaVazia(fp))
103
104
        return 0;
105
       *valor = fp->dados[0].info;
106
      *pri = fp->dados[0].prio;
107
       return 1;
108
    }
109
110 void imprime(FilaP *fp) {
      if (fp == NULL)
111
112
        return;
113
      if (estaVazia(fp)) {
114
        printf("Fila Vazia!\n");
115
        return;
116
      }
117
      printf("Elementos:\n");
118
      int i;
      for (i = 0; i < fp->qtd; i++)
119
120
        printf("[%d, %d] (%d) -- ", fp->dados[i].prio, fp->dados[i].info, i);
121
      printf("\n");
122 }
roteiro_6/2-2.c
 1 #include "./fila_priori_heap.h"
 2
 3
    #define MAX OPTIONS 8
 4
 5
    enum options {
 6
      CRIAR_FILA = 0,
 7
      INSERIR ITEM,
 8
      SHOW INICIO,
 9
      REMOVER ITEM,
      IMPRIMIR,
 10
11
      SHOW TAMANHO,
12
      DESTRUIR,
13
      SAIR,
    };
14
15
16
    int get_option() {
17
      int opt = -1;
      do {
18
19
        printf("\t0peracoes\n");
        printf("[%d] Criar fila, ", CRIAR FILA);
20
21
        printf("[%d] Inserir um item pela prioridade, ", INSERIR ITEM);
        printf("[%d] Ver o início da Fila, ", SHOW INICIO);
22
23
        printf("[%d] Remover um item, \n", REMOVER ITEM);
24
        printf("[%d] Imprimir a Fila, ", IMPRIMIR);
25
        printf("[%d] Mostrar o tamanho da Fila, ", SHOW_TAMANHO);
26
        printf("[%d] Destruir a Fila, ", DESTRUIR);
27
        printf("[%d] Sair do programa \n", SAIR);
28
29
        printf("\nInsira a opção desejada: ");
30
        scanf("%d", &opt);
         printf("\n");
31
32
        if (opt < 0 || opt >= MAX_OPTIONS) {
33
           printf("Opção escolhida inválida!\n");
34
35
36
37
       } while (opt < 0 || opt >= MAX_OPTIONS);
```

```
38
39
       return opt;
40
    }
41
42
    int get_valor(char *msg) {
       int value;
43
44
       printf("%s\n", msg);
45
46
       scanf("%d", &value);
47
48
       return value;
49
    }
50
51
    int main() {
52
       int opt, valor, priori;
53
54
       FilaP *fila priori = NULL;
55
56
       do {
57
         opt = get option();
58
59
         switch (opt) {
60
         case CRIAR FILA:
           printf("Executando comando...\n");
61
62
63
           if (fila_priori == NULL) {
64
             fila_priori = criaFila();
65
           } else {
             destroiFila(&fila_priori);
66
67
             fila priori = NULL;
68
             fila priori = criaFila();
69
70
           printf("Fila de prioridade criada com sucesso!\n");
71
72
           break;
73
         case INSERIR ITEM:
74
           printf("Executando comando...\n");
75
76
           if (fila_priori == NULL) {
77
             printf("Fila de prioridade não instanciado impossivel realizar "
78
                     "operação..\n");
79
             break;
80
           }
81
           if (!inserirPrio(fila priori, get valor("Valor a inserir:"),
82
                             get_valor("Insira a prioridade: "))) {
83
84
             printf("Falha ao inserir na Fila de prioridade!\n");
85
86
87
           printf("Item inserido com sucesso\n");
88
89
           break;
90
91
         case SHOW INICIO:
           printf("Executando comando...\n");
92
93
94
           if (fila priori == NULL) {
95
             printf("Fila de prioridade não instanciado impossivel realizar "
                     "operação..\n");
96
97
             break;
98
           }
99
           if (!verInicio(fila_priori, &valor, &priori)) {
100
101
             printf("Erro ao ver inicio do Fila de prioridade\n");
102
             break;
```

```
5/3/25, 12:00 PM
                                                                Roteiro 6
  103
  104
  105
             printf("Item [valor = %d, prioridade = %d]\n", valor, priori);
  106
             break;
  107
  108
           case REMOVER ITEM:
             printf("Executando comando...\n");
  109
  110
  111
             if (fila priori == NULL) {
  112
               printf("Fila de prioridade não instanciado impossivel realizar "
  113
                       "operação..\n");
  114
               break;
  115
             }
  116
  117
             if (!removeInicio(fila_priori)) {
  118
               printf("Erro ao executar comando\n");
  119
  120
             printf("Removido primeiro item da Fila de prioridade com sucesso!\n");
  121
  122
  123
             break:
  124
           case IMPRIMIR:
  125
  126
             printf("Executando comando...\n");
  127
  128
             if (fila_priori == NULL) {
  129
               printf("Fila de prioridade não instanciado impossivel realizar "
  130
                       "operação..\n");
  131
               break:
  132
             }
  133
  134
             imprime(fila priori);
  135
  136
             break;
           case SHOW TAMANHO:
  137
  138
             printf("Executando comando...\n");
  139
             if (fila_priori == NULL) {
  140
  141
               printf("Fila de prioridade não instanciado impossivel realizar "
  142
                       "operação..\n");
  143
               break:
  144
             }
  145
  146
             printf("Tamanho da fila = %d\n", tamanhoFila(fila priori));
  147
  148
             break;
  149
  150
           case DESTRUIR:
  151
             printf("Executando comando...\n");
  152
  153
             destroiFila(&fila priori);
  154
             fila priori = NULL;
  155
  156
             printf("Fila de prioridade destruida com sucesso!\n");
  157
  158
             break;
  159
           case SAIR:
  160
             printf("Finalizando programa! Até mais!\n");
  161
             break;
  162
  163
  164
         } while (opt != SAIR);
  165
  166
         return 0;
  167 }
```

Saída do terminal:

```
~/Documents/UFSJ-Graduacao/UFSJ-2025_1/Lab_Prog_2/roteiro_6 on mainxxx 25-05-03
L.: ./2-2.out
           Operacoes
operacoes
[0] Criar fila, [1] Inserir um item pela prioridade, [2] Ver o início da Fila, [3] Remover um item,
[4] Imprimir a Fila, [5] Mostrar o tamanho da Fila, [6] Destruir a Fila, [7] Sair do programa
Insira a opção desejada: Θ
Executando comando..
Fila de prioridade criada com sucesso!
Operacoes
[0] Criar fila, [1] Inserir um item pela prioridade, [2] Ver o início da Fila, [3] Remover um item,
[4] Imprimir a Fila, [5] Mostrar o tamanho da Fila, [6] Destruir a Fila, [7] Sair do programa
Insira a opção desejada: 1
Executando comando...
Insira a prioridade:
-
Valor a inserir:
100
Item inserido com sucesso
[0] Criar fila, [1] Inserir um item pela prioridade, [2] Ver o início da Fila, [3] Remover um item,
[4] Imprimir a Fila, [5] Mostrar o tamanho da Fila, [6] Destruir a Fila, [7] Sair do programa
Insira a opcão desejada: 1
Executando comando...
Insira a prioridade:
Valor a inserir:
Item inserido com sucesso
Operacoes
[0] Criar fila, [1] Inserir um item pela prioridade, [2] Ver o início da Fila, [3] Remover um item,
[4] Imprimir a Fila, [5] Mostrar o tamanho da Fila, [6] Destruir a Fila, [7] Sair do programa
Insira a opção desejada: 2
Executando comando...
Item [valor = 2, prioridade = 90]
           Operacoes
operaces
[0] Criar fila, [1] Inserir um item pela prioridade, [2] Ver o início da Fila, [3] Remover um item,
[4] Imprimir a Fila, [5] Mostrar o tamanho da Fila, [6] Destruir a Fila, [7] Sair do programa
Insira a opção desejada: 4
Executando comando...
Elementos:
[90, 2] (0) -- [3, 100] (1) --
Operacoes
[0] Criar fila, [1] Inserir um item pela prioridade, [2] Ver o início da Fila, [3] Remover um item,
[4] Imprimir a Fila, [5] Mostrar o tamanho da Fila, [6] Destruir a Fila, [7] Sair do programa
Insira a opção desejada: 5
Executando comando.
           Operacoes
[0] Criar fila, [1] Inserir um item pela prioridade, [2] Ver o início da Fila, [3] Remover um item,
[4] Imprimir a Fila, [5] Mostrar o tamanho da Fila, [6] Destruir a Fila, [7] Sair do programa
Insira a opção desejada: 6
Executando comando.
Fila de prioridade destruida com sucesso!
Operacoes
[0] Criar fila, [1] Inserir um item pela prioridade, [2] Ver o início da Fila, [3] Remover um item,
[4] Imprimir a Fila, [5] Mostrar o tamanho da Fila, [6] Destruir a Fila, [7] Sair do programa
Insira a opção desejada: 7
Finalizando programa! Até mais!
```