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

```h
#ifndef DEQUE_H
#define DEQUE_H

#include <stdio.h>
#include <stdlib.h>

#define MAX 100

typedef struct {
  int qtd, ini, fim;
  int dados[MAX];
} Deque;

Deque *criaDeque();

void destroiDeque(Deque **dq);

int tamanhoDeque(Deque *dq);

int estaCheio(Deque *dq);

int estaVazio(Deque *dq);

int insereInicio(Deque *dq, int elem);

int insereFim(Deque *dq, int elem);

int removeInicio(Deque *dq);

int removeFim(Deque *dq);

int verInicio(Deque *dq, int *p);

int verFim(Deque *dq, int *p);

void imprime(Deque *dq);

#endif
```

roteiro_6/deque_s_estatico.c

```c
#include "./deque_s_estatico.h"

Deque *criaDeque() {
  Deque *dq;
  dq = (Deque *)malloc(sizeof(Deque));
  if (dq != NULL) {
    dq->qtd = dq->ini = dq->fim = 0;
  }
  return dq;
}
```

```c

void destroiDeque(Deque **dq) {
  if (*dq != NULL) {
    free(*dq);
    *dq = NULL;
  }
}

int tamanhoDeque(Deque *dq) {
  if (dq == NULL)
    return -1;
  return dq->qtd;
}

int estaCheio(Deque *dq) {
  if (dq == NULL)
    return -1;
  return (dq->qtd == MAX);
}

int estaVazio(Deque *dq) {
  if (dq == NULL)
    return -1;
  return (dq->qtd == 0);
}

int insereInicio(Deque *dq, int elem) {
  if (dq == NULL)
    return 0;
  if (estaCheio(dq))
    return 0;
  dq->ini = (dq->ini - 1 < 0 ? MAX - 1 : dq->ini - 1);
  dq->dados[dq->ini] = elem;
  dq->qtd++;
  return 1;
}

int insereFim(Deque *dq, int elem) {
  if (dq == NULL)
    return 0;
  if (estaCheio(dq))
    return 0;
  dq->dados[dq->fim] = elem;
  dq->fim = (dq->fim + 1) % MAX;
  dq->qtd++;
  return 1;
}

int removeInicio(Deque *dq) {
  if (dq == NULL)
    return 0;
  if (estaVazio(dq))
    return 0;
  dq->ini = (dq->ini + 1) % MAX;
  dq->qtd--;
  return 1;
}

int removeFim(Deque *dq) {
  if (dq == NULL)
    return 0;
  if (estaVazio(dq))
    return 0;
  dq->fim = (dq->fim - 1 < 0 ? MAX - 1 : dq->fim - 1);
  dq->qtd--;
```

```c
  return 1;
}

int verInicio(Deque *dq, int *p) {
  if (dq == NULL)
    return 0;
  if (estaVazio(dq))
    return 0;
  *p = dq->dados[dq->ini];
  return 1;
}

int verFim(Deque *dq, int *p) {
  if (dq == NULL)
    return 0;
  if (estaVazio(dq))
    return 0;
  int i = (dq->fim - 1 < 0 ? MAX - 1 : dq->fim - 1);
  *p = dq->dados[i];
  return 1;
}

void imprime(Deque *dq) {
  if (dq == NULL)
    return;
  if (estaVazio(dq)) {
    printf("Deque Vazio!\n");
    return;
  }
  int i = dq->ini;
  printf("Elementos: \n");
  do {
    printf("%d ", dq->dados[i]);
    i = (i + 1) % MAX;
  } while (i != dq->fim);
  // Usar do..while garante a impressao de todos elementos
  // mesmo com a Deque cheia
  printf("\n");
}
```

roteiro_6/1-1.c

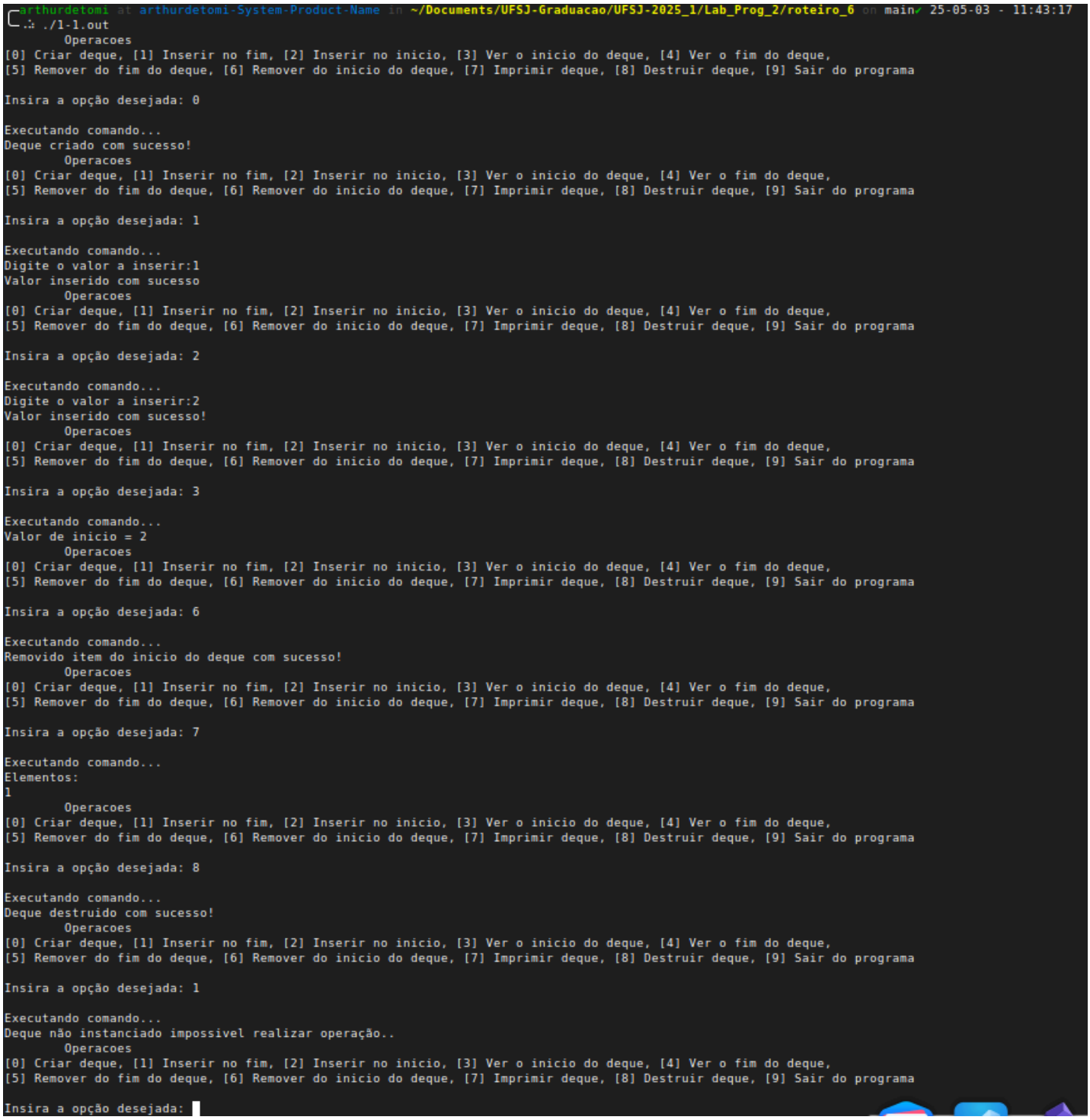
```c
#include "./deque_s_estatico.h"

#define MAX_OPTIONS 10

enum options {
  CRIAR_FILA = 0,
  INSERIR_ITEM,
  INSERIR_INICIO,
  SHOW_INICIO,
  SHOW_FIM,
  REMOVER_ITEM,
  REMOVER_INICIO,
  IMPRIMIR,
  DESTRUIR,
  SAIR,
};

int get_option() {
  int opt = -1;
  do {
    printf("\tOperacoes\n");
    printf("[%d] Criar deque, ", CRIAR_FILA);
    printf("[%d] Inserir no fim, ", INSERIR_ITEM);
    printf("[%d] Inserir no inicio, ", INSERIR_INICIO);
```

```
25      printf("[%d] Ver o inicio do deque, ", SHOW_INICIO);
26      printf("[%d] Ver o fim do deque, ", SHOW_FIM);
27      printf("\n[%d] Remover do fim do deque, ", REMOVER_ITEM);
28      printf("[%d] Remover do inicio do deque, ", REMOVER_INICIO);
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) {
38        printf("Opção escolhida inválida!\n");
39      }
40
41    } while (opt < 0 || opt >= MAX_OPTIONS);
42
43    return opt;
44  }
45
46  int get_valor() {
47    int value;
48
49    printf("Digite o valor a inserir:");
50    scanf("%d", &value);
51
52    return value;
53  }
54
55  int main() {
56    int opt, valor;
57
58    Deque *deque = NULL;
59
60    do {
61      opt = get_option();
62
63      switch (opt) {
64      case CRIAR_FILA:
65        printf("Executando comando...\n");
66
67        if (deque == NULL) {
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())) {
86          printf("Falha ao inserir no fim do deque!\n");
87        }
88
89        printf("Valor inserido com sucesso\n");
```

```
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())) {
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
111       if (deque == NULL) {
112         printf("Deque não instanciado impossivel realizar operação..\n");
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;
123     case SHOW_FIM:
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");
143         break;
144       }
145
146       if (!removeFim(deque)) {
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");
```

```
155
156       if (deque == NULL) {
157         printf("Deque não instanciado impossivel realizar operação..\n");
158         break;
159       }
160
161       if (!removeInicio(deque)) {
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:
169       printf("Executando comando...\n");
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:



1.2) TAD: Deque Duplamente Encadeado

roteiro_6/deque_encadeado.h

```c
#ifndef DEQUE_ENCADEADO
#define DEQUE_ENCADEADO

#include <stdio.h>
#include <stdlib.h>

typedef struct NO {
  int info;
  struct NO *prox;
  struct NO *ant;
} NO;

typedef struct {
  int qtd;
  struct NO *ini;
```

```
  struct NO *fim;
} Deque;

NO *alocarNO();

void liberarNO(NO *q);

Deque *criaDeque();

void destroiDeque(Deque **dq);

int tamanhoDeque(Deque *dq);

int estaVazio(Deque *dq);

int insereInicio(Deque *dq, int elem);

int insereFim(Deque *dq, int elem);

int removeInicio(Deque *dq);

int removeFim(Deque *dq);

int verInicio(Deque *dq, int *p);

int verFim(Deque *dq, int *p);

void imprime(Deque *dq);

#endif
```

roteiro_6/deque_encadeado.c

```
#include "./deque_encadeado.h"

NO *alocarNO() { return (NO *)malloc(sizeof(NO)); }

void liberarNO(NO *q) { free(q); }

Deque *criaDeque() {
  Deque *dq;
  dq = (Deque *)malloc(sizeof(Deque));
  if (dq != NULL) {
    dq->qtd = 0;
    dq->ini = NULL;
    dq->fim = NULL;
  }
  return dq;
}

void destroiDeque(Deque **dq) {
  if (*dq != NULL) {
    NO *aux;
    while ((*dq)->ini != NULL) {
      aux = (*dq)->ini;
      (*dq)->ini = (*dq)->ini->prox;
      liberarNO(aux);
    }
    free(*dq);
    *dq = NULL;
  }
}

int tamanhoDeque(Deque *dq) {
  if (dq == NULL)
    return -1;
```

```c
  return dq->qtd;
}

int estaVazio(Deque *dq) {
  if (dq == NULL)
    return -1;
  return (dq->qtd == 0);
}

int insereInicio(Deque *dq, int elem) {
  if (dq == NULL)
    return 0;
  NO *novo = alocarNO();
  if (novo == NULL)
    return 0;
  novo->info = elem;
  novo->ant = NULL;
  if (estaVazio(dq)) {
    novo->prox = NULL;
    dq->fim = novo;
  } else {
    dq->ini->ant = novo;
    novo->prox = dq->ini;
  }
  dq->ini = novo;
  dq->qtd++;
  return 1;
}

int insereFim(Deque *dq, int elem) {
  if (dq == NULL)
    return 0;
  NO *novo = alocarNO();
  if (novo == NULL)
    return 0;
  novo->info = elem;
  novo->prox = NULL;
  if (estaVazio(dq)) {
    novo->ant = NULL;
    dq->ini = novo;
  } else {
    dq->fim->prox = novo;
    novo->ant = dq->fim;
  }
  dq->fim = novo;
  dq->qtd++;
  return 1;
}

int removeInicio(Deque *dq) {
  if (dq == NULL)
    return 0;
  if (estaVazio(dq))
    return 0;
  NO *aux = dq->ini;
  if (dq->ini == dq->fim) {
    dq->ini = dq->fim = NULL;
  } else {
    dq->ini = dq->ini->prox;
    dq->ini->ant = NULL;
  }
  liberarNO(aux);
  dq->qtd--;
  return 1;
}
```

```c

int removeFim(Deque *dq) {
  if (dq == NULL)
    return 0;
  if (estaVazio(dq))
    return 0;
  NO *aux = dq->fim;
  if (dq->ini == dq->fim) {
    dq->ini = dq->fim = NULL;
  } else {
    dq->fim = dq->fim->ant;
    dq->ini->prox = NULL;
  }
  liberarNO(aux);
  dq->qtd--;
  return 1;
}

int verInicio(Deque *dq, int *p) {
  if (dq == NULL)
    return 0;
  if (estaVazio(dq))
    return 0;
  *p = dq->ini->info;
  return 1;
}

int verFim(Deque *dq, int *p) {
  if (dq == NULL)
    return 0;
  if (estaVazio(dq))
    return 0;
  *p = dq->fim->info;
  return 1;
}

void imprime(Deque *dq) {
  if (dq == NULL)
    return;
  if (estaVazio(dq)) {
    printf("Deque Vazio!\n");
    return;
  }
  NO *aux = dq->ini;
  printf("Elementos:\n");
  while (aux != NULL) {
    printf("%d ", aux->info);
    aux = aux->prox;
  }
  printf("\n");
}
```

roteiro_6/1-2.c

```c
#include "./deque_encadeado.h"

#define MAX_OPTIONS 10

enum options {
  CRIAR_FILA = 0,
  INSERIR_ITEM,
  INSERIR_INICIO,
  SHOW_INICIO,
  SHOW_FIM,
  REMOVER_ITEM,
  REMOVER_INICIO,
```

```
  IMPRIMIR,
  DESTRUIR,
  SAIR,
};

int get_option() {
  int opt = -1;
  do {
    printf("\tOperacoes\n");
    printf("[%d] Criar deque, ", CRIAR_FILA);
    printf("[%d] Inserir no fim, ", INSERIR_ITEM);
    printf("[%d] Inserir no inicio, ", INSERIR_INICIO);
    printf("[%d] Ver o inicio do deque, ", SHOW_INICIO);
    printf("[%d] Ver o fim do deque, ", SHOW_FIM);
    printf("\n[%d] Remover do fim do deque, ", REMOVER_ITEM);
    printf("[%d] Remover do inicio do deque, ", REMOVER_INICIO);
    printf("[%d] Imprimir deque, ", IMPRIMIR);
    printf("[%d] Destruir deque, ", DESTRUIR);
    printf("[%d] Sair do programa \n", SAIR);

    printf("\nInsira a opção desejada: ");
    scanf("%d", &opt);
    printf("\n");

    if (opt < 0 || opt >= MAX_OPTIONS) {
      printf("Opção escolhida inválida!\n");
    }

  } while (opt < 0 || opt >= MAX_OPTIONS);

  return opt;
}

int get_valor() {
  int value;

  printf("Digite o valor a inserir:");
  scanf("%d", &value);

  return value;
}

int main() {
  int opt, valor;

  Deque *deque = NULL;

  do {
    opt = get_option();

    switch (opt) {
    case CRIAR_FILA:
      printf("Executando comando...\n");

      if (deque == NULL) {
        deque = criaDeque();
      } else {
        destroiDeque(&deque);
        deque = NULL;
        deque = criaDeque();
      }

      printf("Deque criado com sucesso!\n");
      break;
    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())) {
86          printf("Falha ao inserir no fim do deque!\n");
87        }
88
89        printf("Valor inserido com sucesso\n");
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())) {
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
111       if (deque == NULL) {
112         printf("Deque não instanciado impossivel realizar operação..\n");
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;
123     case SHOW_FIM:
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");
```

```
143         break;
144       }
145
146       if (!removeFim(deque)) {
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");
155
156       if (deque == NULL) {
157         printf("Deque não instanciado impossivel realizar operação..\n");
158         break;
159       }
160
161       if (!removeInicio(deque)) {
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:
169       printf("Executando comando...\n");
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:

```
└─$ ./1-2.out
        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: 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: 1

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: 2

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:
1 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: 3

Executando comando...
Valor de inicio = 1
        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: 5

Executando comando...
Removido item do fim 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: 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

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: 9
```

2.1) TAD: Fila de Prioridades Simplesmente Encadeada

roteiro_6/fila_prioridade.h

```c
#ifndef FILA_PRIORIDADE
#define FILA_PRIORIDADE

#include <stdio.h>
#include <stdlib.h>

typedef struct NO {
  int info, prio;
  struct NO *prox;
```

```
} NO;

typedef struct NO *FilaP;

NO *alocarNO();

void liberarNO(NO *q);

FilaP *criaFila();

int estaVazia(FilaP *fp);

int inserirPrio(FilaP *fp, int elem, int pri);

int removeInicio(FilaP *fp);

int verInicio(FilaP *fp, int *p, int *pri);

void imprime(FilaP *fp);

void destroiFila(FilaP **fp);

int tamanhoFila(FilaP *fp);

#endif
```

roteiro_6/fila_prioridade.c

```
#include "./fila_prioridade.h"

NO *alocarNO() { return (NO *)malloc(sizeof(NO)); }

void liberarNO(NO *q) { free(q); }

FilaP *criaFila() {
  FilaP *fp;
  fp = (FilaP *)malloc(sizeof(FilaP));
  if (fp != NULL)
    *fp = NULL;
  return fp;
}

int estaVazia(FilaP *fp) {
  if (fp == NULL)
    return -1;
  return ((*fp) == NULL);
}

int inserirPrio(FilaP *fp, int elem, int pri) {
  if (fp == NULL)
    return 0;
  NO *novo = alocarNO();
  if (novo == NULL)
    return 0;

  novo->info = elem;
  novo->prio = pri;

  if (estaVazia(fp)) {
    novo->prox = *fp;
    *fp = novo;
  } else {
    NO *aux, *ant;
    ant = NULL;
    aux = *fp; // Inicio
    while (aux != NULL && aux->prio >= novo->prio) {
```

```
      ant = aux;
      aux = aux->prox;
    }
    if (ant == NULL) {
      novo->prox = *fp;
      *fp = novo;
    } else {
      novo->prox = ant->prox;
      ant->prox = novo;
    }
  }
  return 1;
}

int removeInicio(FilaP *fp) {
  if (fp == NULL)
    return 0;
  if (estaVazia(fp))
    return 0;
  NO *aux = *fp;
  *fp = aux->prox;
  liberarNO(aux);
  return 1;
}

int verInicio(FilaP *fp, int *p, int *pri) {
  if (fp == NULL)
    return 0;
  if (estaVazia(fp))
    return 0;
  *p = (*fp)->info;
  *pri = (*fp)->prio;
  return 1;
}

void imprime(FilaP *fp) {
  if (fp == NULL)
    return;
  if (estaVazia(fp)) {
    printf("Fila Vazia!\n");
    return;
  }
  NO *aux = *fp;
  while (aux != NULL) {
    printf("[%d, %d] ", aux->prio, aux->info);
    aux = aux->prox;
  }
  printf("\n");
}

void destroiFila(FilaP **fp) {
  if (*fp != NULL) {
    NO *aux;
    while ((**fp) != NULL) {
      aux = **fp;
      **fp = (**fp)->prox;
      liberarNO(aux);
    }
    free(*fp);
    *fp = NULL;
  }
}

int tamanhoFila(FilaP *fp) {
  if (fp == NULL)
```

```c
    return -1;

  int t = 0;
  NO *i = *fp;
  while (i != NULL) {
    t++;
    i = i->prox;
  }

  return t;
}
```

roteiro_6/2-1.c

```c
#include "fila_prioridade.h"

#define MAX_OPTIONS 8

enum options {
  CRIAR_FILA = 0,
  INSERIR_ITEM,
  SHOW_INICIO,
  REMOVER_ITEM,
  IMPRIMIR,
  SHOW_TAMANHO,
  DESTRUIR,
  SAIR,
};

int get_option() {
  int opt = -1;
  do {
    printf("\tOperacoes\n");
    printf("[%d] Criar fila, ", CRIAR_FILA);
    printf("[%d] Inserir um item pela prioridade, ", INSERIR_ITEM);
    printf("[%d] Ver o início da Fila, ", SHOW_INICIO);
    printf("[%d] Remover um item, \n", REMOVER_ITEM);
    printf("[%d] Imprimir a Fila, ", IMPRIMIR);
    printf("[%d] Mostrar o tamanho da Fila, ", SHOW_TAMANHO);
    printf("[%d] Destruir a Fila, ", DESTRUIR);
    printf("[%d] Sair do programa \n", SAIR);

    printf("\nInsira a opção desejada: ");
    scanf("%d", &opt);
    printf("\n");

    if (opt < 0 || opt >= MAX_OPTIONS) {
      printf("Opção escolhida inválida!\n");
    }

  } while (opt < 0 || opt >= MAX_OPTIONS);

  return opt;
}

int get_valor(char *msg) {
  int value;

  printf("%s\n", msg);
  scanf("%d", &value);

  return value;
}

int main() {
  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
71        printf("Fila de prioridade criada com sucesso!\n");
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
82        if (!inserirPrio(fila_priori, get_valor("Valor a inserir:"),
83                         get_valor("Insira a prioridade: "))) {
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:
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)) {
101         printf("Erro ao ver inicio do Fila de prioridade\n");
102         break;
103       }
104
105       printf("Item [valor = %d, prioridade = %d]\n", valor, priori);
106       break;
107
108     case REMOVER_ITEM:
109       printf("Executando comando...\n");
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
121       printf("Removido primeiro item da Fila de prioridade com sucesso!\n");
122
123       break;
124
125     case IMPRIMIR:
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;
137     case SHOW_TAMANHO:
138       printf("Executando comando...\n");
139
140       if (fila_priori == NULL) {
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:

```
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: 0

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:
10
Valor a inserir:
1
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: 1

Executando comando...
Insira a prioridade:
100
Valor a inserir:
0
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: 4

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
[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 = 0, 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!
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 = 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

```c
#ifndef FILA_PRIORI_HEAP
#define FILA_PRIORI_HEAP

#include <stdio.h>
#include <stdlib.h>

#define MAX 100

typedef struct NO {
  int info, prio;
```

```
} NO;

typedef struct {
  int qtd;
  NO dados[MAX];
} FilaP;

FilaP *criaFila();

void destroiFila(FilaP **fp);

int tamanhoFila(FilaP *fp);

int estaCheia(FilaP *fp);

int estaVazia(FilaP *fp);

void trocaNO(NO *a, NO *b);

void ajustaHeapInsere(FilaP *fp, int filho);

int inserirPrio(FilaP *fp, int elem, int pri);

void ajustaHeapRemove(FilaP *fp, int pai);

int removeInicio(FilaP *fp);

int verInicio(FilaP *fp, int *valor, int *pri);

void imprime(FilaP *fp);

#endif
```

roteiro_6/fila_priori_heap.c

```
#include "./fila_priori_heap.h"

FilaP *criaFila() {
  FilaP *fp;
  fp = (FilaP *)malloc(sizeof(FilaP));
  if (fp != NULL)
    fp->qtd = 0;
  return fp;
}

void destroiFila(FilaP **fp) {
  if (*fp != NULL) {
    free(*fp);
    *fp = NULL;
  }
}

int tamanhoFila(FilaP *fp) {
  if (fp == NULL)
    return -1;
  return fp->qtd;
}

int estaCheia(FilaP *fp) {
  if (fp == NULL)
    return -1;
  return (fp->qtd == MAX);
}

int estaVazia(FilaP *fp) {
  if (fp == NULL)
```

```
32      return -1;
33    return (fp->qtd == 0);
34  }
35
36  void trocaNO(NO *a, NO *b) {
37    NO temp;
38    temp.info = a->info;
39    temp.prio = a->prio;
40    a->info = b->info;
41    a->prio = b->prio;
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;
49    int prioFilho = fp->dados[filho].prio;
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;
55      prioFilho = fp->dados[filho].prio;
56    }
57  }
58
59  int inserirPrio(FilaP *fp, int elem, int pri) {
60    if (fp == NULL)
61      return 0;
62    if (estaCheia(fp))
63      return 0;
64    fp->dados[fp->qtd].info = elem;
65    fp->dados[fp->qtd].prio = pri;
66    ajustaHeapInsere(fp, fp->qtd);
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        break;
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;
96    ajustaHeapRemove(fp, 0);
```

```c
  return 1;
}

int verInicio(FilaP *fp, int *valor, int *pri) {
  if (fp == NULL)
    return 0;
  if (estaVazia(fp))
    return 0;
  *valor = fp->dados[0].info;
  *pri = fp->dados[0].prio;
  return 1;
}

void imprime(FilaP *fp) {
  if (fp == NULL)
    return;
  if (estaVazia(fp)) {
    printf("Fila Vazia!\n");
    return;
  }
  printf("Elementos:\n");
  int i;
  for (i = 0; i < fp->qtd; i++)
    printf("[%d, %d] (%d) -- ", fp->dados[i].prio, fp->dados[i].info, i);
  printf("\n");
}
```

roteiro_6/2-2.c

```c
#include "./fila_priori_heap.h"

#define MAX_OPTIONS 8

enum options {
  CRIAR_FILA = 0,
  INSERIR_ITEM,
  SHOW_INICIO,
  REMOVER_ITEM,
  IMPRIMIR,
  SHOW_TAMANHO,
  DESTRUIR,
  SAIR,
};

int get_option() {
  int opt = -1;
  do {
    printf("\tOperacoes\n");
    printf("[%d] Criar fila, ", CRIAR_FILA);
    printf("[%d] Inserir um item pela prioridade, ", INSERIR_ITEM);
    printf("[%d] Ver o início da Fila, ", SHOW_INICIO);
    printf("[%d] Remover um item, \n", REMOVER_ITEM);
    printf("[%d] Imprimir a Fila, ", IMPRIMIR);
    printf("[%d] Mostrar o tamanho da Fila, ", SHOW_TAMANHO);
    printf("[%d] Destruir a Fila, ", DESTRUIR);
    printf("[%d] Sair do programa \n", SAIR);

    printf("\nInsira a opção desejada: ");
    scanf("%d", &opt);
    printf("\n");

    if (opt < 0 || opt >= MAX_OPTIONS) {
      printf("Opção escolhida inválida!\n");
    }

  } while (opt < 0 || opt >= MAX_OPTIONS);
```

```c

  return opt;
}

int get_valor(char *msg) {
  int value;

  printf("%s\n", msg);
  scanf("%d", &value);

  return value;
}

int main() {
  int opt, valor, priori;

  FilaP *fila_priori = NULL;

  do {
    opt = get_option();

    switch (opt) {
    case CRIAR_FILA:
      printf("Executando comando...\n");

      if (fila_priori == NULL) {
        fila_priori = criaFila();
      } else {
        destroiFila(&fila_priori);
        fila_priori = NULL;
        fila_priori = criaFila();
      }

      printf("Fila de prioridade criada com sucesso!\n");
      break;
    case INSERIR_ITEM:
      printf("Executando comando...\n");

      if (fila_priori == NULL) {
        printf("Fila de prioridade não instanciado impossivel realizar "
               "operação..\n");
        break;
      }

      if (!inserirPrio(fila_priori, get_valor("Valor a inserir:"),
                       get_valor("Insira a prioridade: "))) {
        printf("Falha ao inserir na Fila de prioridade!\n");
      }

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

      break;

    case SHOW_INICIO:
      printf("Executando comando...\n");

      if (fila_priori == NULL) {
        printf("Fila de prioridade não instanciado impossivel realizar "
               "operação..\n");
        break;
      }

      if (!verInicio(fila_priori, &valor, &priori)) {
        printf("Erro ao ver inicio do Fila de prioridade\n");
        break;
```

```
      }

      printf("Item [valor = %d, prioridade = %d]\n", valor, priori);
      break;

    case REMOVER_ITEM:
      printf("Executando comando...\n");

      if (fila_priori == NULL) {
        printf("Fila de prioridade não instanciado impossivel realizar "
               "operação..\n");
        break;
      }

      if (!removeInicio(fila_priori)) {
        printf("Erro ao executar comando\n");
      }

      printf("Removido primeiro item da Fila de prioridade com sucesso!\n");

      break;

    case IMPRIMIR:
      printf("Executando comando...\n");

      if (fila_priori == NULL) {
        printf("Fila de prioridade não instanciado impossivel realizar "
               "operação..\n");
        break;
      }

      imprime(fila_priori);

      break;
    case SHOW_TAMANHO:
      printf("Executando comando...\n");

      if (fila_priori == NULL) {
        printf("Fila de prioridade não instanciado impossivel realizar "
               "operação..\n");
        break;
      }

      printf("Tamanho da fila = %d\n", tamanhoFila(fila_priori));

      break;

    case DESTRUIR:
      printf("Executando comando...\n");

      destroiFila(&fila_priori);
      fila_priori = NULL;

      printf("Fila de prioridade destruida com sucesso!\n");

      break;
    case SAIR:
      printf("Finalizando programa! Até mais!\n");
      break;
    }

  } while (opt != SAIR);

  return 0;
}
```

Saída do terminal:

```
arthurdetomi at arthurdetomi-System-Product-Name in ~/Documents/UFSJ-Graduacao/UFSJ-2025_1/Lab_Prog_2/roteiro_6 on mainxxx 25-05-03
./2-2.out
        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: 0

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:
3
Valor a inserir:
100
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: 1

Executando comando...
Insira a prioridade:
90
Valor a inserir:
2
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
[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...
Tamanho da fila = 2
        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!
```