



Universidade Federal de São João del Rei  
Departamento de Ciência da Computação  
Curso de Ciência da Computação

## Roteiro 5

Rodrigo José Zonzin  
212050002

### 1 Fila

#### 1.1

```
1 #include "fila.h"
2
3 Fila* cria_fila(){
4     Fila *f = (Fila*)malloc(sizeof(Fila*));
5     f->elementos = (int*)malloc(sizeof(int)*50);
6     f->n = 0;
7
8     return f;
9 }
10
11 void enfileira(Fila *f, int a){
12     f->elementos[f->n] = a;
13     f->n = f->n + 1;
14 }
15
16 void desenfileira(Fila *f){
17     for(int i = 0; i < (f->n)-1; i++){
18         f->elementos[i] = f->elementos[i+1];
19     }
20     f->n = f->n - 1;
21 }
22
23 void imprime_fila(Fila *f){
24     printf("[");
25     for(int i = 0; i < (f->n)-1; i++){
26         printf("%d ", f->elementos[i]);
27     }
28     printf("%d]\n", f->elementos[f->n - 1]);
29 }
30
31 int inicio_fila(Fila *f){
32     return f->elementos[0];
33 }
34
35 void destroiFila(Fila *f){
36     free(f->elementos);
37     free(f);
38 }
39
```

```

40 void clearTerminal(){
41     #ifdef _WIN32
42         system("cls"); // Limpa o terminal no Windows
43     #else
44         system("clear"); // Limpa o terminal no Unix/Linux
45     #endif
46 }

```

../ex1/fila.c

```

1 #define _FILA_H
2
3 #include <stdio.h>
4 #include <stdlib.h>
5
6
7 struct fila{
8     int *elementos;
9     int n;
10 };
11 typedef struct fila Fila;
12
13 Fila* cria_fila();
14 void enfilera(Fila*, int);
15 void desenfilera(Fila*);
16 void imprime_fila(Fila*);
17 void destroiFila(Fila*);

```

../ex1/fila.h

```

1 #include "fila.h"
2
3 int main(){
4     int chamada = 10;
5     int elemento;
6     Fila *f;
7
8     while(chamada != 7){
9         printf("\n\n1-Cria Fila\t2-Enfilera\t3-VerInicio\n");
10        printf("4-Desenfilera\t5-Imprimir\t6-Destruir\n");
11        scanf("%d", &chamada);
12
13
14        if(chamada == 1){
15            f = cria_fila();
16            continue;
17        }
18
19        if(chamada == 2){
20            printf("Qual elemento?\n");
21            scanf("%d", &elemento);
22            enfilera(f, elemento);
23            continue;
24        }
25
26        if(chamada == 3){
27            printf("%d\n", inicio_fila(f));
28            continue;
29        }
30

```

```

31         if(chamada == 4){
32             desenfilera(f);
33             continue;
34         }
35
36         if(chamada==5){
37             imprime_fila(f);
38             continue;
39         }
40
41         if(chamada == 6){
42             destroiFila(f);
43             break;
44         }
45     }
46     printf("E eh isso\n");
47     return 0;
48 }

```

../ex1/main.c

## 1.2

```

1  #include "fila.h"
2
3  Fila* cria_fila(){
4      Fila*f=(Fila*)malloc(sizeof(Fila));
5      f->qtd=0;
6      f->ini= NULL;
7      f->fim= NULL;
8      return f;
9  }
10
11 void enfilera(Fila*f, int a){
12     NO* novo=(NO*)malloc(sizeof(NO));
13
14     novo->info = a;
15     novo->prox = NULL;
16
17     if(f->qtd == 0){
18         f->ini = novo;
19         f->fim = novo;
20     }
21     else{
22         f->fim->prox=novo;
23         f->fim=novo;
24     }
25
26     f->qtd++;
27 }
28
29 void desenfilera(Fila*f){
30     if(f->qtd==0){
31         return;
32     }
33
34     NO* temp = f->ini;
35     f->ini=f->ini->prox;
36     free(temp);
37     f->qtd--;

```

```

38 }
39
40 void imprime_fila(Fila*f){
41     printf("[");
42     NO*atual=f->ini;
43
44     while(atual!=NULL){
45         if(atual->prox == NULL){
46             printf("%d]\n", atual->info);
47             break;
48         }
49         printf("%d ",atual->info);
50         atual=atual->prox;
51     }
52 }
53
54 int inicio_fila(Fila*f){
55     if(f->qtd==0){
56         printf("Lista vazia.\n");
57         return -1;
58     }
59     return f->ini->info;
60 }
61
62 void destroiFila(Fila*f){
63     NO*atual=f->ini;
64     while(atual!=NULL){
65         NO*temp=atual;
66         atual=atual->prox;
67         free(temp);
68     }
69     free(f);
70 }

```

../ex2/fila.c

```

1 #define _FILA_H
2
3 #include <stdlib.h>
4 #include <stdio.h>
5
6 typedef struct NO{
7     int info ;
8     struct NO* prox ;
9 }NO;
10
11 typedef struct fila{
12     int qtd;
13     struct NO* ini;
14     struct NO* fim;
15 }Fila;
16
17 Fila* cria_fila();
18 void enfilera(Fila*, int);
19 void desenfilera(Fila*);
20 void imprime_fila(Fila*);
21 int inicio_fila(Fila*);
22 void destroiFila(Fila*);

```

../ex2/fila.h

```

1 #include "fila.h"
2
3 int main(){
4     Fila *f = cria_fila();
5
6     enfilera(f, 00);
7     enfilera(f, 01);
8     enfilera(f, 10);
9     enfilera(f, 11);
10
11     imprime_fila(f);
12
13     desenfilera(f);
14     imprime_fila(f);
15
16
17
18     return 0;
19 }

```

../ex2/main.c

## 2 Pilha

### 2.1

```

1 /*
2     Criar pilha;
3     Empilhar um item;
4     Ver o topo da pilha.
5     Desempilhar um item;
6     Imprimir a pilha;
7     Destruir a pilha;
8     Sair;
9 */
10 #include "pilha.h"
11
12 #include <stdio.h>
13 #include <stdlib.h>
14
15 Pilha* cria_pilha(int tamanho){
16     Pilha* p = (Pilha*)malloc(sizeof(Pilha));
17     p->elementos = (int*)malloc(sizeof(int) * tamanho);
18     p->n = 0;
19     return p;
20 }
21
22 void empilhar(Pilha* p, int valor){
23     p->elementos[p->n] = valor;
24     p->n = p->n + 1;
25 }
26
27 int ver_topo(Pilha* p){
28     if (p->n > 0){
29         return p->elementos[p->n - 1];
30     }
31     else{
32         printf("A pilha esta vazia.\n");
33         return -1;

```

```

34     }
35 }
36
37 void desempilhar(Pilha* p){
38     if (p->n > 0) {
39         p->n--;
40     } else {
41         printf("A pilha esta vazia.\n");
42     }
43 }
44
45 void imprimir_pilha(Pilha* p) {
46     printf("[");
47     for (int i = 0; i < p->n - 1; i++) {
48         printf("%d, ", p->elementos[i]);
49     }
50
51     if (p->n > 0) {
52         printf("%d", p->elementos[p->n - 1]);
53     }
54
55     printf("]\n");
56 }
57
58 void destruir_pilha(Pilha* p) {
59     free(p->elementos);
60     free(p);
61 }

```

../ex3/pilha.c

```

1 #define _PILHA_H
2
3 struct pilha{
4     int *elementos;
5     int n;
6 };
7 typedef struct pilha Pilha;

```

../ex3/pilha.h

```

1 #include "pilha.h"
2
3 int main(){
4     int tamanho = 50;
5     Pilha* pilha = cria_pilha(tamanho);
6
7     empilhar(pilha, 10);
8     empilhar(pilha, 20);
9     empilhar(pilha, 30);
10    imprimir_pilha(pilha);
11
12    printf("Topo: %d\n", ver_topo(pilha));
13
14    desempilhar(pilha);
15    desempilhar(pilha);
16
17    imprimir_pilha(pilha);
18
19    destruir_pilha(pilha);

```

```

20
21     return 0;
22 }

```

../ex3/main.c

## 2.2

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #include "pilha.h"
5
6  Pilha* cria_pilha(){
7      Pilha* p = (Pilha*)malloc(sizeof(Pilha));
8      p->qtd = 0;
9      p->topo = NULL;
10     return p;
11 }
12
13 void empilhar(Pilha* p, int valor){
14     NO* novo = (NO*)malloc(sizeof(NO));
15     novo->info = valor;
16     novo->prox = p->topo;
17     p->topo = novo;
18     p->qtd++;
19 }
20
21 int ver_topo(Pilha* p){
22     if(p->qtd > 0){
23         return p->topo->info;
24     }
25     else{
26         printf("A pilha esta vazia.\n");
27         return -1;
28     }
29 }
30
31 void desempilhar(Pilha* p){
32     if (p->qtd > 0){
33         NO* temp = p->topo;
34         p->topo = p->topo->prox;
35         free(temp);
36         p->qtd--;
37     }
38     else{
39         printf("A pilha esta vazia.\n");
40     }
41 }
42
43 void imprimir_pilha(Pilha* p){
44     printf("Pilha: [");
45     NO* atual = p->topo;
46
47     while(atual != NULL){
48         printf("%d", atual->info);
49         if(atual->prox != NULL) {
50             printf(", ");
51         }
52         atual = atual->prox;

```

```

53     }
54     printf("]\n");
55 }
56
57 void destruir_pilha(Pilha* p){
58     while(p->topo != NULL){
59         NO* temp = p->topo;
60         p->topo = p->topo->prox;
61         free(temp);
62     }
63     free(p);
64 }

```

../ex4/pilha.c

```

1 #define _PILHA_H
2
3
4 typedef struct NO{
5     int info ;
6     struct NO* prox ;
7 }NO;
8
9 typedef struct{
10     int qtd;
11     struct NO* topo ;
12 }Pilha ;

```

../ex4/pilha.h

```

1 #include "pilha.h"
2
3 int main(){
4     Pilha* pilha = cria_pilha();
5
6     empilhar(pilha, 10);
7     empilhar(pilha, 20);
8     empilhar(pilha, 30);
9
10    printf("Topo da pilha: %d\n", ver_topo(pilha));
11
12    desempilhar(pilha);
13
14    imprimir_pilha(pilha);
15    destruir_pilha(pilha);
16
17    return 0;
18 }

```

../ex4/main.c

### 3 Código

<https://github.com/RodrigoZonzin/labaeds2>