

## ROTEIRO 5 – LUCAS EDUARDO LEITE COSTA

### 1.1

```
questão-1 > C main.c > main()
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include "fila.h"
4
5 int main(){
6     Fila *F;
7     int escolha, elem;
8     do{
9         scanf("%d", &escolha);
10        switch(escolha){
11            case 1:
12                F = criaFila();
13                break;
14            case 2:
15                scanf("%d", &elem);
16                enfileirar(F,elem);
17                break;
18            case 3:
19                printf("Inicio da fila: %d\n", verInicio(F));
20                break;
21            case 4:
22                desenfileirar(F);
23                break;
24            case 5:
25                imprime(F);
26                break;
27            case 6:
28                destroiFila(F);
29                break;
30            case 7:
31                break;
32            default:
33                printf("Opção inválida\n");
34        }
35    }while(escolha!=7);
36    return 0;
37 }
38
39
```

```
questão-1 > C fila.c > ...
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include "fila.h"
4
5 Fila* criaFila(){
6     Fila * fi;
7     fi = (Fila*)malloc(sizeof(Fila));
8     if(fi != NULL )
9         fi->qtd = fi->ini = fi->fim = 0;
10    return fi;
11 }
12
13 void destroiFila(Fila *fi){
14     if(fi != NULL)
15         free(fi);
16 }
17
18 int tamanhoFila(Fila *fi){
19     if(fi == NULL )
20         return -1;
21     return fi->qtd;
22 }
23
24 int estaCheia( Fila *fi){
25     if(fi == NULL )
26         return -1;
27     return (fi->qtd == MAX );
28 }
29
30 int estaVazia(Fila *fi){
31     if(fi == NULL )
32         return -1;
33     return (fi->qtd == 0);
34 }
35
36 int enfileirar(Fila *fi , int elem){
37     if(fi == NULL )
38         return 0;
39     if(estaCheia(fi))
40         return 0;
41     fi->dados[fi->fim] = elem ;
42     fi->fim = (fi->fim+1) % MAX ;
43     fi->qtd ++;
44     return 1;
45 }
46
47 int desenfileirar(Fila *fi){
48     if(fi == NULL )
49         return 0;
50     if(estaVazia( fi))
51         return 0;
52     fi->ini = (fi-> ini +1) % MAX ;
53     fi->qtd--;
54     return 1;
55 }
56
57 int verInicio(Fila * fi){
58     if(fi == NULL)
59         return 0;
60     if(estaVazia(fi))
61         return 0;
62     return fi->dados[fi->ini];
63 }
64
65 void imprime ( Fila * fi){
66     if(fi == NULL)
67         return;
68     if(estaVazia( fi)){
69         printf ("Fila Vazia!\n");
70         return;
71     }
72     int i = fi->ini;
73     printf ("Elementos: \n");
74     do{
75         printf ("%d ", fi -> dados [i] );
76         i = (i + 1) % MAX;
77     }while(i != fi->fim );
78     printf ("\n");
79 }
```

```

gcc main.c fila.c -o main
[lucascosta@fedora questão-1]$ ./main
1
2
4
2
5
2
4
2
6
3
Início da fila: 4
4
3
Início da fila: 5
5
Elementos:
5 4 6
6
7
[lucascosta@fedora questão-1]$ █

```

1.2

questão-1.1 > C main.c > ...	questão-1.1 > C fila.h > ...
<pre> 1 #include &lt;stdio.h&gt; 2 #include &lt;stdlib.h&gt; 3 #include "fila.h" 4 5 int main(){ 6     Fila *F; 7     int escolha, elem; 8     do{ 9         scanf("%d", &amp;escolha); 10        switch(escolha){ 11            case 1: 12                F = criaFila(); 13                break; 14            case 2: 15                scanf("%d", &amp;elem); 16                enfileirar(F,elem); 17                break; 18            case 3: 19                printf("Início da fila: %d\n", verInicio(F)); 20                break; 21            case 4: 22                desenfileirar(F); 23                break; 24            case 5: 25                imprime(F); 26                break; 27            case 6: 28                destroiFila(F); 29                break; 30            case 7: 31                break; 32            default: 33                printf("Opção inválida\n"); 34        } 35    }while(escolha!=7); 36    return 0; 37 } </pre>	<pre> 1 #ifndef FILA_H 2 #define FILA_H 3 #include &lt;stdio.h&gt; 4 #include &lt;stdlib.h&gt; 5 6 typedef struct NO{ 7     int info ; 8     struct NO* prox ; 9 }NO; 10 11 typedef struct{ 12     int qtd; 13     struct NO* ini; 14     struct NO* fim; 15 }Fila ; 16 17 Fila* criaFila(); 18 void destroiFila(Fila *fi); 19 int tamanhoFila(Fila *fi); 20 int estaVazia(Fila *fi); 21 int enfileirar(Fila *fi , int elem); 22 int desenfileirar(Fila *fi); 23 int verInicio(Fila * fi); 24 void imprime ( Fila * fi); 25 26 #endif </pre>

```

[lucascosta@fedora questão-1.1]$ ./main
1
2
5
2
4
2
6
2
8
3
Início da fila: 5
4
3
Início da fila: 4
5
Elementos: 4 6 8
6
7
[lucascosta@fedora questão-1.1]$ █

```

```

questão-1.1 > C fila.c > ...
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "fila.h"
4
5 Fila* criaFila(){
6     Fila *fi = (Fila *)malloc(sizeof(Fila));
7     if (fi != NULL)
8         fi->ini = fi->fim = NULL;
9     return fi;
10 }
11
12 void destroiFila(Fila *fi){
13     NO* aux = fi->ini;
14     NO* aux2;
15     while(aux!=NULL){
16         aux2 = aux->prox;
17         free(aux);
18         aux = aux2;
19     }
20     free(fi);
21 }
22 int tamanhoFila(Fila *fi){
23     int tam = 0;
24     if(fi == NULL)
25         return tam;
26     NO* aux = fi->ini;
27     while(aux != NULL){
28         tam++;
29         aux = aux->prox;
30     }
31     return tam;
32 }
33
34 int estaVazia(Fila *fi){
35     return (fi->ini == NULL);
36 }
37 int enfileirar(Fila *fi, int elem){
38     NO* n = (NO*)malloc(sizeof(NO));
39     n->info = elem;
40     n->prox = NULL;
41     if(!estaVazia(fi))
42         fi->fim->prox = n;
43     else{
44         fi->ini = n;
45     }
46     fi->fim = n;
47
48 }

```

```

questão-1.1 > C fila.c > ...
48 }
49 int desenfileirar(Fila *fi){
50     NO* aux;
51     if(fi == NULL)
52         return 0;
53     if(estaVazia(fi)){
54         printf("Fila vazia!\n");
55         return 0;
56     }
57     aux = fi->ini;
58     fi->ini = aux->prox;
59     if(fi->ini == NULL)
60         fi->fim = NULL;
61     free(aux);
62     return 1;
63 }
64 int verInicio(Fila * fi){
65     if(fi == NULL)
66         return 0;
67
68     if(estaVazia(fi))
69         return 0;
70     int elem;
71     elem = fi->ini->info;
72     return elem;
73 }
74
75 void imprime(Fila * fi){
76     if(fi == NULL)
77         return;
78     if(estaVazia(fi)){
79         printf("Fila vazia!\n");
80         return;
81     }
82     printf("Elementos: ");
83     NO* aux = fi->ini;
84     while(aux!=NULL){
85         printf("%d ", aux->info);
86         aux = aux->prox;
87     }
88     printf("\n");
89
90 }

```

questão-2 > C main.c > ...

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "pilha.h"
4
5 int main(){
6     Pilha* pi;
7     int escolha, elem;
8     do{
9         scanf("%d", &escolha);
10        switch(escolha){
11            case 1:
12                pi = criaPilha();
13                break;
14            case 2:
15                scanf("%d", &elem);
16                empilhar(pi, elem);
17                break;
18            case 3:
19                printf("TOPO DA PILHA: %d\n", verTopo(pi));
20                break;
21            case 4:
22                desempilhar(pi);
23                break;
24            case 5:
25                imprime(pi);
26                break;
27            case 6:
28                destroiPilha(pi);
29                break;
30            case 7:
31                break;
32            default:
33                printf("Opção inválida\n");
34        }
35    }while(escolha!=7);
36
37    return 0;
38 }
39
```

questão-2 > C pilha.h > ...

```
1 #ifndef PILHA_H
2 #define PILHA_H
3 #include <stdio.h>
4 #include <stdlib.h>
5 #define MAX 100
6
7 typedef struct {
8     int topo;
9     int dados [MAX];
10 }Pilha;
11
12 Pilha* criaPilha();
13 void destroiPilha(Pilha* pi);
14 int tamanhoPilha(Pilha* pi);
15 int estaCheia(Pilha* pi);
16 int estaVazia(Pilha* pi);
17 int empilhar(Pilha* pi , int elem);
18 int desempilhar(Pilha* pi);
19 int verTopo(Pilha* pi);
20 void imprime(Pilha* pi);
21
22 #endif
```

questão-2 > C pilha.c > ...

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "pilha.h"
4
5 Pilha* criaPilha(){
6     Pilha * pi;
7     pi = (Pilha*) malloc(sizeof(Pilha ));
8     if(pi != NULL)
9         pi->topo = 0;
10    return pi;
11 }
12
13 void destroiPilha(Pilha* pi){
14     if(pi != NULL )
15         free(pi);
16 }
17
18 int tamanhoPilha(Pilha* pi){
19     if(pi==NULL)
20         return -1;
21     return pi->topo;
22 }
23
24 int estaCheia(Pilha* pi){
25     if(pi==NULL)
26         return -1;
27     return (pi->topo==MAX);
28 }
29
30 int estaVazia(Pilha* pi){
31     if(pi==NULL)
32         return -1;
33     return (pi->topo==0);
34 }
35
36 int empilhar(Pilha* pi , int elem){
37     if(pi==NULL)
38         return 0;
39     if(estaCheia(pi))
40         return 0;
41     pi->dados[pi->topo] = elem;
42     pi->topo++;
43     return 1;
44 }
45
```

questão-2 > C pilha.c > ...

```
42 int desempilhar(Pilha* pi){
43     if(pi==NULL)
44         return 0;
45     if(estaVazia(pi))
46         return 0;
47     pi->topo--;
48     return 1;
49 }
50
51 int verTopo(Pilha* pi){
52     if(pi==NULL)
53         return 0;
54     if(estaVazia(pi))
55         return 0;
56     return pi->dados[pi->topo-1];
57 }
58
59 void imprime(Pilha* pi){
60     if(pi == NULL )
61         return;
62     if( estaVazia(pi)){
63         printf ("Pilha Vazia!\n");
64         return;
65     }
66     printf("Elementos: ");
67     int i;
68     for(i=pi->topo-1; i >=0; i--){
69         printf ("%d ", pi->dados [i]) ;
70     }
71     printf ("\n");
72 }
```

```
[lucascosta@fedora questão-2]$ ./main
1
2
3
2
5
2
4
2
6
3
TOPO DA PILHA: 6
4
5
Elementos: 4 5 3
6
7
[lucascosta@fedora questão-2]$
```

## 2.2

```
questão-2.1 > C main.c > ...
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "pilha.h"
4
5 int main(){
6     Pilha* pi;
7     int escolha, elem;
8     do{
9         scanf("%d", &escolha);
10        switch(escolha){
11            case 1:
12                pi = criaPilha();
13                break;
14            case 2:
15                scanf("%d", &elem);
16                empilhar(pi,elem);
17                break;
18            case 3:
19                printf("TOPO DA PILHA: %d\n", verTopo(pi));
20                break;
21            case 4:
22                desempilhar(pi);
23                break;
24            case 5:
25                imprime(pi);
26                break;
27            case 6:
28                destroiPilha(pi);
29                break;
30            case 7:
31                break;
32            default:
33                printf("Opção inválida\n");
34        }
35    }while(escolha!=7);
36
37    return 0;
38 }

questão-2.1 > C pilha.h > ...
1 #ifndef PILHA_H
2 #define PILHA_H
3 #include <stdio.h>
4 #include <stdlib.h>
5
6
7 typedef struct NO{
8     int info;
9     struct NO* prox ;
10 }NO;
11
12 typedef struct {
13     int qtd;
14     struct NO* topo ;
15 }Pilha;
16
17 Pilha* criaPilha();
18 void destroiPilha(Pilha* pi);
19 int tamanhoPilha(Pilha* pi);
20 int estaVazia(Pilha* pi);
21 int empilhar(Pilha* pi , int elem);
22 int desempilhar(Pilha* pi);
23 int verTopo(Pilha* pi);
24 void imprime(Pilha* pi);
25
26 #endif
```

questão-2.1 > C pilha.c > ...

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "pilha.h"
4
5 Pilha* criaPilha(){
6     Pilha* pi = (Pilha*)malloc(sizeof(Pilha));
7     pi->topo = NULL;
8     return pi;
9 }
10 void destroiPilha(Pilha* pi){
11     NO* aux;
12     NO* q = pi->topo;
13     while(q!=NULL){
14         aux = q->prox;
15         free(q);
16         q = aux;
17     }
18     free(pi);
19 }
20 int tamanhoPilha(Pilha* pi){
21     int tam = 0;
22     if(pi == NULL)
23         return tam;
24     NO* aux = pi->topo;
25     while(aux!=NULL){
26         tam++;
27         aux = aux->prox;
28     }
29     return tam;
30 }
31 int estaVazia(Pilha* pi){
32     return (pi->topo==NULL);
33 }
34 int empilhar(Pilha* pi, int elem){
35     NO* aux = (NO*)malloc(sizeof(NO));
36     if(aux==NULL)
37         return 0;
38     aux->info = elem;
39     aux->prox = pi->topo;
40     pi->topo = aux;
41     return 1;
42 }
```

questão-2.1 > C pilha.c > ...

```
43 int desempilhar(Pilha* pi){
44     NO* aux;
45     if(estaVazia(pi)){
46         printf("Pilha vazia!\n");
47         return 0;
48     }
49     aux = pi->topo;
50     pi->topo = aux->prox;
51     free(aux);
52     return 1;
53 }
54 int verTopo(Pilha* pi){
55     return pi->topo->info;
56 }
57 void imprime(Pilha* pi){
58     if(pi == NULL)
59         return;
60     if(estaVazia(pi)){
61         printf("Pilha vazia!\n");
62         return;
63     }
64     printf("Elementos: ");
65     NO* aux = pi->topo;
66     while(aux!=NULL){
67         printf("%d ", aux->info);
68         aux = aux->prox;
69     }
70     printf("\n");
71 }
```

[lucascosta@fedora questão-2.1]\$ ./main

```
1
2
5
2
6
2
3
3
TOPO DA PILHA: 3
4
5
Elementos: 6 5
6
7
[lucascosta@fedora questão-2.1]$
```