

Atividade 8 - Filas e Matrizes Lineares (T1)

23 de outubro de 2025

Problema A

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef struct No {
5     struct No *anterior;
6     struct No *proximo;
7     int valor;
8 } No;
9
10 typedef struct Fila {
11     struct No *inicio;
12     struct No *fim;
13     int tamanho;
14 } Fila;
15
16 Fila* criaFila(void){
17     Fila *fila = (Fila*) malloc (sizeof(Fila));
18
19     if(!fila){
20         printf("Erro! Nao foi possivel alocar memoria para a fila.\n");
21         return NULL;
22     }
23
24     fila->inicio = NULL;
25     fila->fim = NULL;
26     fila->tamanho = 0;
27
28     return fila;
29 }
30
31 int filaExiste(Fila *fila){
32     return fila != NULL;
33 }
34
35 int filaVazia(Fila *fila){
36     if(!filaExiste(fila)){
37         printf("Erro! A fila nao existe.\n");
38         return 1;
39     }
40     return fila->tamanho == 0;
41 }
42
```

```
43 void insereFila(Fila *fila, int valor){
44     if(!filaExiste(fila)){
45         printf("Erro! A fila nao existe.\n");
46         return;
47     }
48
49     No *no = (No*) malloc (sizeof(No));
50
51     if(!no){
52         printf("Erro! Nao foi possivel alocar o no na memoria.\n");
53         return;
54     }
55
56     no->valor = valor;
57     no->proximo = NULL;
58
59     if(filaVazia(fila)){
60         fila->inicio = no;
61         no->anterior = NULL;
62     } else if (fila->tamanho > 1){
63         no->anterior = fila->fim;
64         fila->fim->proximo = no;
65     } else {
66         no->anterior = fila->inicio;
67         fila->inicio->proximo = no;
68     }
69
70     fila->fim = no;
71     fila->tamanho += 1;
72 }
73
74 int removeFila(Fila* fila){
75     if(filaVazia(fila)){
76         printf("Erro! A fila esta vazia.\n");
77         return -1;
78     }
79
80     int valor = fila->inicio->valor;
81
82     if(fila->tamanho > 1){
83         fila->inicio = fila->inicio->proximo;
84         free(fila->inicio->anterior);
85         fila->inicio->anterior = NULL;
86     } else {
87         free(fila->inicio);
88         fila->inicio = NULL;
89         fila->fim = NULL;
90     }
91
92     fila->tamanho -= 1;
93     return valor;
94 }
95
96 void destroiFila(Fila *fila){
97     if(!filaExiste(fila)) return;
98
99     while(fila->tamanho){
100         removeFila(fila);
```

```
101     }
102
103     free(fila);
104 }
105
106 int main(){
107     int n;
108
109     Fila *fila = criaFila();
110
111     while(1){
112         scanf("%d", &n);
113         if(n == 0) break;
114
115         for(int i = 1; i <= n; ++i){
116             insereFila(fila, i);
117         }
118
119         int rounds = n - 1;
120         int* discarded_cards = (int*) malloc (sizeof(int) * rounds);
121         int j = 0;
122         while(rounds--){
123             discarded_cards[j++] = removeFila(fila);
124             int mover = removeFila(fila);
125             insereFila(fila, mover);
126         }
127
128         int remaining_card = removeFila(fila);
129
130         printf("Discarded cards:");
131         for(int i = 0; i < j; ++i){
132             if(i == 0){
133                 printf(" %d", discarded_cards[i]);
134             } else {
135                 printf(", %d", discarded_cards[i]);
136             }
137         }
138         printf("\n");
139
140         printf("Remaining card: %d\n", remaining_card);
141
142         free(discarded_cards);
143     }
144
145     destroiFila(fila);
146
147     return 0;
148 }
```

Problema B

```
1 // Funcao para atualizar um elemento na posicao (linha, coluna)
2 void atualizaElemento(Matriz *m, int linha, int coluna, int valor) {
3     m->matriz[m->num_colunas*linha + coluna] = valor;
4 }
5
6 // Funcao para recuperar um elemento na posicao (linha, coluna)
7 int recuperaElemento(Matriz *m, int linha, int coluna) {
8     return m->matriz[m->num_colunas*linha + coluna]
9 }
```

Problema C

```
1 // Funcao para atualizar um elemento na posicao (linha, coluna)
2 void atualizaElemento(Matriz *m, int linha, int coluna, int valor) {
3     m->matriz[linha * (linha + 1)/2 + coluna] = valor;
4 }
5
6 // Funcao para recuperar um elemento na posicao (linha, coluna)
7 int recuperaElemento(Matriz *m, int linha, int coluna) {
8     if (linha < coluna) {
9         return 0;
10    }
11
12    return m->matriz[linha * (linha + 1)/2 + coluna];
13 }
```

Problema D

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef struct No {
5     struct No *anterior;
6     struct No *proximo;
7     int valor;
8 } No;
9
10 typedef struct Fila {
11     struct No *inicio;
12     struct No *fim;
13     int tamanho;
14 } Fila;
15
16 Fila* criaFila(void){
17     Fila *fila = (Fila*) malloc (sizeof(Fila));
18
19     if(!fila){
20         printf("Erro! Nao foi possivel alocar memoria para a fila.\n");
21         return NULL;
22     }
23
24     fila->inicio = NULL;
25     fila->fim = NULL;
26     fila->tamanho = 0;
27
28     return fila;
29 }
30
31 int filaExiste(Fila *fila){
32     return fila != NULL;
33 }
34
35 int filaVazia(Fila *fila){
36     if(!filaExiste(fila)){
37         printf("Erro! A fila nao existe.\n");
38         return 1;
39     }
40     return fila->tamanho == 0;
41 }
42
43 void insereFila(Fila *fila, int valor){
44     if(!filaExiste(fila)){
45         printf("Erro! A fila nao existe.\n");
46         return;
47     }
48
49     No *no = (No*) malloc (sizeof(No));
50
51     if(!no){
52         printf("Erro! Nao foi possivel alocar o no na memoria.\n");
53         return;
54     }
55
56     no->valor = valor;
```

```
57     no->proximo = NULL;
58
59     if(filaVazia(fila)){
60         fila->inicio = no;
61         no->anterior = NULL;
62     } else if (fila->tamanho > 1){
63         no->anterior = fila->fim;
64         fila->fim->proximo = no;
65     } else {
66         no->anterior = fila->inicio;
67         fila->inicio->proximo = no;
68     }
69
70     fila->fim = no;
71     fila->tamanho += 1;
72 }
73
74 int removeFila(Fila* fila){
75     if(filaVazia(fila)){
76         printf("Erro! A fila esta vazia.\n");
77         return -1;
78     }
79
80     int valor = fila->inicio->valor;
81
82     if(fila->tamanho > 1){
83         fila->inicio = fila->inicio->proximo;
84         free(fila->inicio->anterior);
85         fila->inicio->anterior = NULL;
86     } else {
87         free(fila->inicio);
88         fila->inicio = NULL;
89         fila->fim = NULL;
90     }
91
92     fila->tamanho -= 1;
93     return valor;
94 }
95
96 void destroiFila(Fila *fila){
97     if(!filaExiste(fila)) return;
98
99     while(fila->tamanho){
100         removeFila(fila);
101     }
102
103     free(fila);
104 }
105
106 int main(){
107     int n, k;
108     scanf("%d %d", &n, &k);
109
110     Fila *fila = criaFila();
111
112     for(int i = 0; i < n; ++i){
113         int x;
114         scanf("%d", &x);
```

```
115     insereFila(fila, x);
116 }
117
118 int ans = 0;
119 while(fila->tamanho >= 3){
120     int sum = 0;
121     No* aux = fila->inicio;
122
123     for(int i = 0; i < 3; ++i){
124         sum += aux->valor;
125         aux = aux->proximo;
126     }
127
128     if(sum == k){
129         ans = 1;
130         break;
131     }
132
133     removeFila(fila);
134 }
135
136 if(ans == 0){
137     printf("Nao\n");
138 } else {
139     printf("Sim\n");
140 }
141
142 destroiFila(fila);
143 return 0;
144 }
```