JOÃO ANTÔNIO SANTOS CARVALHO - 192050052

EX 1.1

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

#define MAX 100
typedef struct{
    int qtd, ini, fim;
    int dados[MAX];
}Fila;


Fila* criaFila(){
    Fila* fi;
    fi = (Fila*) malloc (sizeof(Fila));
    if(fi != NULL){
        fi->qtd = fi->ini = fi->fim = 0;
    }
    return fi;
}

void destroiFila(Fila *fi){
    if(fi != NULL)
        free(fi);
}

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

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

int estaVazia(Fila *fi){
    if(fi == NULL)
        return -1;
    return (fi->qtd == 0);
}


int enfileirar(Fila* fi, int elem){
    if(fi == NULL) return 0;
    if(estaCheia(fi)) return 0;
    fi->dados[fi->fim] = elem;
```

```
    fi->fim = (fi->fim+1) % MAX;
    fi->qtd++;
    return 1;
}

int desenfileirar(Fila* fi){
    if(fi == NULL) return 0;
    if(estaVazia(fi)) return 0;
    fi->ini = (fi->ini+1) % MAX;
    fi->qtd--;
    return 1;
}

int verInicio(Fila* fi, int* p){
  if(fi == NULL) return 0;
  if(estaVazia(fi)) return 0;
  *p = fi->dados[fi->ini];
  return 1;
}

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

int main() {
    Fila *fila = NULL;
    int opcao, elemento, inicio;

    do {
        printf("\nMenu:\n");
        printf("1 - Criar fila\n");
        printf("2 - Enfileirar um item\n");
        printf("3 - Ver o início da fila\n");
        printf("4 - Desenfileirar um item\n");
        printf("5 - Imprimir a fila\n");
        printf("6 - Destruir a fila\n");
        printf("7 - Sair\n");
        printf("Escolha uma opção: ");
        scanf("%d", &opcao);
```

```
switch (opcao) {
    case 1:
        if (fila == NULL) {
            fila = criaFila();
            printf("Fila criada com sucesso.\n");
        } else {
            printf("A fila já foi criada.\n");
        }
        break;

    case 2:
        if (fila != NULL) {
            printf("Digite o elemento a ser enfileirado: ");
            scanf("%d", &elemento);
            if (enfileirar(fila, elemento)) {
                printf("Elemento enfileirado com sucesso.\n");
            } else {
                printf("Erro ao enfileirar o elemento.\n");
            }
        } else {
            printf("A fila não foi criada ainda.\n");
        }
        break;

    case 3:
        if (fila != NULL) {
            if (verInicio(fila, &inicio)) {
                printf("O início da fila é: %d\n", inicio);
            } else {
                printf("A fila está vazia.\n");
            }
        } else {
            printf("A fila não foi criada ainda.\n");
        }
        break;

    case 4:
        if (fila != NULL) {
            if (desenfileirar(fila)) {
                printf("Elemento desenfileirado com sucesso.\n");
            } else {
                printf("A fila está vazia.\n");
            }
        } else {
            printf("A fila não foi criada ainda.\n");
        }
        break;

    case 5:
        if (fila != NULL) {
            imprime(fila);
```

```
            } else {
                printf("A fila não foi criada ainda.\n");
            }
            break;

        case 6:
            if (fila != NULL) {
                destroiFila(fila);
                fila = NULL;
                printf("Fila destruída com sucesso.\n");
            } else {
                printf("A fila não foi criada ainda.\n");
            }
            break;

        case 7:
            printf("Saindo do programa.\n");
            if (fila != NULL) {
                destroiFila(fila);
            }
            break;

        default:
            printf("Opção inválida. Tente novamente.\n");
            break;
    }
  } while (opcao != 7);

  return 0;
}
```

```
scjoaoantonio@scjoaoantonio:~/Área de Trabalho/AEDS2/Aulas Práticas/Lista5/Respostas$ gcc -o ex11 ./ex11.c
scjoaoantonio@scjoaoantonio:~/Área de Trabalho/AEDS2/Aulas Práticas/Lista5/Respostas$ ./ex11

Menu:
1 - Criar fila
2 - Enfileirar um item
3 - Ver o início da fila
4 - Desenfileirar um item
5 - Imprimir a fila
6 - Destruir a fila
7 - Sair
Escolha uma opção: 1
Fila criada com sucesso.
```

EX 1.2

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

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

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

```
} Fila;

Fila* criaFila() {
    Fila* fila = (Fila*)malloc(sizeof(Fila));
    if (fila != NULL) {
        fila->qtd = 0;
        fila->ini = fila->fim = NULL;
    }
    return fila;
}

void destroiFila(Fila* fila) {
    if (fila != NULL) {
        while (fila->ini != NULL) {
            NO* aux = fila->ini;
            fila->ini = fila->ini->prox;
            free(aux);
        }
        free(fila);
    }
}

int estaVazia(Fila* fila) {
    return (fila == NULL || fila->qtd == 0);
}

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

int enfileirar(Fila* fila, int elemento) {
    if (fila == NULL) {
        return 0;
    }
    NO* novo = (NO*)malloc(sizeof(NO));
    if (novo == NULL) {
        return 0;
    }
    novo->info = elemento;
    novo->prox = NULL;
    if (fila->fim == NULL) {
        fila->ini = fila->fim = novo;
    } else {
        fila->fim->prox = novo;
        fila->fim = novo;
    }
    fila->qtd++;
    return 1;
}
```

```
int desenfileirar(Fila* fila) {
    if (fila == NULL || fila->ini == NULL) {
        return 0;
    }
    NO* aux = fila->ini;
    fila->ini = fila->ini->prox;
    free(aux);
    fila->qtd--;
    if (fila->ini == NULL) {
        fila->fim = NULL;
    }
    return 1;
}

int verInicio(Fila* fila, int* p) {
    if (fila == NULL || fila->ini == NULL) {
        return 0;
    }
    *p = fila->ini->info;
    return 1;
}

void imprime(Fila* fila) {
    if (fila == NULL || fila->qtd == 0) {
        printf("Fila Vazia!\n");
        return;
    }
    NO* atual = fila->ini;
    printf("Elementos: \n");
    while (atual != NULL) {
        printf("%d ", atual->info);
        atual = atual->prox;
    }
    printf("\n");
}

int main() {
    Fila* fila = NULL;
    int opcao, elemento, inicio;

    do {
        printf("\nMenu:\n");
        printf("1 - Criar fila\n");
        printf("2 - Enfileirar um item\n");
        printf("3 - Ver o início da fila\n");
        printf("4 - Desenfileirar um item\n");
        printf("5 - Imprimir a fila\n");
        printf("6 - Destruir a fila\n");
        printf("7 - Sair\n");
        printf("Escolha uma opção: ");
        scanf("%d", &opcao);
```

```
switch (opcao) {
    case 1:
        if (fila == NULL) {
            fila = criaFila();
            printf("Fila criada com sucesso.\n");
        } else {
            printf("A fila já foi criada.\n");
        }
        break;

    case 2:
        if (fila != NULL) {
            printf("Digite o elemento a ser enfileirado: ");
            scanf("%d", &elemento);
            if (enfileirar(fila, elemento)) {
                printf("Elemento enfileirado com sucesso.\n");
            } else {
                printf("Erro ao enfileirar o elemento.\n");
            }
        } else {
            printf("A fila não foi criada ainda.\n");
        }
        break;

    case 3:
        if (fila != NULL) {
            if (verInicio(fila, &inicio)) {
                printf("O início da fila é: %d\n", inicio);
            } else {
                printf("A fila está vazia.\n");
            }
        } else {
            printf("A fila não foi criada ainda.\n");
        }
        break;

    case 4:
        if (fila != NULL) {
            if (desenfileirar(fila)) {
                printf("Elemento desenfileirado com sucesso.\n");
            } else {
                printf("A fila está vazia.\n");
            }
        } else {
            printf("A fila não foi criada ainda.\n");
        }
        break;

    case 5:
        if (fila != NULL) {
            imprime(fila);
```

```
            } else {
                printf("A fila não foi criada ainda.\n");
            }
            break;

        case 6:
            if (fila != NULL) {
                destroiFila(fila);
                fila = NULL;
                printf("Fila destruída com sucesso.\n");
            } else {
                printf("A fila não foi criada ainda.\n");
            }
            break;

        case 7:
            printf("Saindo do programa.\n");
            if (fila != NULL) {
                destroiFila(fila);
            }
            break;

        default:
            printf("Opção inválida. Tente novamente.\n");
            break;
      }
   } while (opcao != 7);

   return 0;
}
```

```
scjoaoantonio@scjoaoantonio:~/Área de Trabalho/AEDS2/Aulas Práticas/Lista5/Respostas$ gcc -o ex12 ./ex12.c
scjoaoantonio@scjoaoantonio:~/Área de Trabalho/AEDS2/Aulas Práticas/Lista5/Respostas$ ./ex12

Menu:
1 - Criar fila
2 - Enfileirar um item
3 - Ver o início da fila
4 - Desenfileirar um item
5 - Imprimir a fila
6 - Destruir a fila
7 - Sair
Escolha uma opção: 1
Fila criada com sucesso.
```

EX 2.1

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

#define MAX 100

typedef struct{
   int topo;
   int dados[MAX];
}Pilha;


Pilha* criaPilha(){
```

```
    Pilha* pi;
    pi = (Pilha*) malloc (sizeof(Pilha));
    if(pi != NULL){
        pi->topo = 0;
    }
    return pi;
}

void destroiPilha(Pilha *pi){
    if(pi != NULL)
        free(pi);
}

int tamanhoPilha(Pilha *pi){
    if(pi == NULL)
        return -1;
    return pi->topo;
}

int estaCheia(Pilha *pi){
    if(pi == NULL)
        return -1;
    return (pi->topo == MAX);
}

int estaVazia(Pilha *pi){
    if(pi == NULL)
        return -1;
    return (pi->topo == 0);
}


int empilhar(Pilha* pi, int elem){
    if(pi == NULL) return 0;
    if(estaCheia(pi)) return 0;
    pi->dados[pi->topo] = elem;
    pi->topo++;
    return 1;
}

int desempilhar(Pilha* pi){
    if(pi == NULL) return 0;
    if(estaVazia(pi)) return 0;
    pi->topo--;
    return 1;
}

int verTopo(Pilha* pi, int* p){
  if(pi == NULL) return 0;
  if(estaVazia(pi)) return 0;
  *p = pi->dados[pi->topo-1];
  return 1;
```

```
}

void imprime(Pilha* pi){
  if(pi == NULL) return;
  if(estaVazia(pi)){
    printf("Pilha Vazia!\n");
    return;
  }
  printf("Elementos: \n");
  int i;
  for(i=pi->topo-1; i>=0; i--)
    printf("%d ", pi->dados[i]);
  printf("\n");
}

int main() {
    Pilha *pilha = NULL;
    int opcao, elemento, topo;

    do {
        printf("\nMenu:\n");
        printf("1 - Criar pilha\n");
        printf("2 - Empilhar um item\n");
        printf("3 - Ver o topo da pilha\n");
        printf("4 - Desempilhar um item\n");
        printf("5 - Imprimir a pilha\n");
        printf("6 - Destruir a pilha\n");
        printf("7 - Sair\n");
        printf("Escolha uma opção: ");
        scanf("%d", &opcao);

        switch (opcao) {
            case 1:
                if (pilha == NULL) {
                    pilha = criaPilha();
                    printf("Pilha criada com sucesso.\n");
                } else {
                    printf("A pilha já foi criada.\n");
                }
                break;

            case 2:
                if (pilha != NULL) {
                    printf("Digite o elemento a ser empilhado: ");
                    scanf("%d", &elemento);
                    if (empilhar(pilha, elemento)) {
                        printf("Elemento empilhado com sucesso.\n");
                    } else {
                        printf("Erro ao empilhar o elemento.\n");
                    }
                } else {
                    printf("A pilha não foi criada ainda.\n");
```

```
    }
    break;

case 3:
    if (pilha != NULL) {
        if (verTopo(pilha, &topo)) {
            printf("O topo da pilha é: %d\n", topo);
        } else {
            printf("A pilha está vazia.\n");
        }
    } else {
        printf("A pilha não foi criada ainda.\n");
    }
    break;

case 4:
    if (pilha != NULL) {
        if (desempilhar(pilha)) {
            printf("Elemento desempilhado com sucesso.\n");
        } else {
            printf("A pilha está vazia.\n");
        }
    } else {
        printf("A pilha não foi criada ainda.\n");
    }
    break;

case 5:
    if (pilha != NULL) {
        imprime(pilha);
    } else {
        printf("A pilha não foi criada ainda.\n");
    }
    break;

case 6:
    if (pilha != NULL) {
        destroiPilha(pilha);
        pilha = NULL;
        printf("Pilha destruída com sucesso.\n");
    } else {
        printf("A pilha não foi criada ainda.\n");
    }
    break;

case 7:
    printf("Saindo do programa.\n");
    if (pilha != NULL) {
        destroiPilha(pilha);
    }
    break;
```

```
            default:
                printf("Opção inválida. Tente novamente.\n");
                break;
        }
    } while (opcao != 7);

    return 0;
}
```

```
3 - Ver o topo da pilha
4 - Desempilhar um item
5 - Imprimir a pilha
6 - Destruir a pilha
7 - Sair
Escolha uma opção: 1
Pilha criada com sucesso.

Menu:
1 - Criar pilha
2 - Empilhar um item
3 - Ver o topo da pilha
4 - Desempilhar um item
5 - Imprimir a pilha
6 - Destruir a pilha
7 - Sair
Escolha uma opção: 2
Digite o elemento a ser empilhado: 10
Elemento empilhado com sucesso.
```

EX 2.2

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

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

typedef struct {
    int qtd;
    NO* topo;
} Pilha;

Pilha* criaPilha() {
    Pilha* pilha = (Pilha*)malloc(sizeof(Pilha));
    if (pilha != NULL) {
        pilha->qtd = 0;
        pilha->topo = NULL;
    }
    return pilha;
}

void destroiPilha(Pilha* pilha) {
    if (pilha != NULL) {
```

```
        while (pilha->topo != NULL) {
            NO* aux = pilha->topo;
            pilha->topo = pilha->topo->prox;
            free(aux);
        }
        free(pilha);
    }
}

int estaVazia(Pilha* pilha) {
    return (pilha == NULL || pilha->qtd == 0);
}

int tamanhoPilha(Pilha* pilha) {
    if (pilha == NULL) {
        return -1;
    }
    return pilha->qtd;
}

int empilhar(Pilha* pilha, int elemento) {
    if (pilha == NULL) {
        return 0;
    }
    NO* novo = (NO*)malloc(sizeof(NO));
    if (novo == NULL) {
        return 0;
    }
    novo->info = elemento;
    novo->prox = pilha->topo;
    pilha->topo = novo;
    pilha->qtd++;
    return 1;
}

int desempilhar(Pilha* pilha) {
    if (pilha == NULL || pilha->topo == NULL) {
        return 0;
    }
    NO* aux = pilha->topo;
    pilha->topo = pilha->topo->prox;
    free(aux);
    pilha->qtd--;
    return 1;
}

int verTopo(Pilha* pilha, int* p) {
    if (pilha == NULL || pilha->topo == NULL) {
        return 0;
    }
    *p = pilha->topo->info;
    return 1;
```

```
}

void imprime(Pilha* pilha) {
    if (pilha == NULL || pilha->qtd == 0) {
        printf("Pilha Vazia!\n");
        return;
    }
    NO* atual = pilha->topo;
    printf("Elementos: \n");
    while (atual != NULL) {
        printf("%d ", atual->info);
        atual = atual->prox;
    }
    printf("\n");
}

int main() {
    Pilha* pilha = NULL;
    int opcao, elemento, topo;

    do {
        printf("\nMenu:\n");
        printf("1 - Criar pilha\n");
        printf("2 - Empilhar um item\n");
        printf("3 - Ver o topo da pilha\n");
        printf("4 - Desempilhar um item\n");
        printf("5 - Imprimir a pilha\n");
        printf("6 - Destruir a pilha\n");
        printf("7 - Sair\n");
        printf("Escolha uma opção: ");
        scanf("%d", &opcao);

        switch (opcao) {
            case 1:
                if (pilha == NULL) {
                    pilha = criaPilha();
                    printf("Pilha criada com sucesso.\n");
                } else {
                    printf("A pilha já foi criada.\n");
                }
                break;

            case 2:
                if (pilha != NULL) {
                    printf("Digite o elemento a ser empilhado: ");
                    scanf("%d", &elemento);
                    if (empilhar(pilha, elemento)) {
                        printf("Elemento empilhado com sucesso.\n");
                    } else {
                        printf("Erro ao empilhar o elemento.\n");
                    }
                } else {
```

```
            printf("A pilha não foi criada ainda.\n");
        }
        break;

    case 3:
        if (pilha != NULL) {
            if (verTopo(pilha, &topo)) {
                printf("O topo da pilha é: %d\n", topo);
            } else {
                printf("A pilha está vazia.\n");
            }
        } else {
            printf("A pilha não foi criada ainda.\n");
        }
        break;

    case 4:
        if (pilha != NULL) {
            if (desempilhar(pilha)) {
                printf("Elemento desempilhado com sucesso.\n");
            } else {
                printf("A pilha está vazia.\n");
            }
        } else {
            printf("A pilha não foi criada ainda.\n");
        }
        break;

    case 5:
        if (pilha != NULL) {
            imprime(pilha);
        } else {
            printf("A pilha não foi criada ainda.\n");
        }
        break;

    case 6:
        if (pilha != NULL) {
            destroiPilha(pilha);
            pilha = NULL;
            printf("Pilha destruída com sucesso.\n");
        } else {
            printf("A pilha não foi criada ainda.\n");
        }
        break;

    case 7:
        printf("Saindo do programa.\n");
        if (pilha != NULL) {
            destroiPilha(pilha);
        }
        break;
```

```
            default:
                printf("Opção inválida. Tente novamente.\n");
                break;
        }
    } while (opcao != 7);

    return 0;
}
```

```
scjoaoantonio@scjoaoantonio:~/Área de Trabalho/AEDS2/Aulas Práticas/Lista5/Respostas$ gcc -o ex22 ./ex22.c
scjoaoantonio@scjoaoantonio:~/Área de Trabalho/AEDS2/Aulas Práticas/Lista5/Respostas$ ./ex22

Menu:
1 - Criar pilha
2 - Empilhar um item
3 - Ver o topo da pilha
4 - Desempilhar um item
5 - Imprimir a pilha
6 - Destruir a pilha
7 - Sair
Escolha uma opção: 1
Pilha criada com sucesso.

Menu:
1 - Criar pilha
2 - Empilhar um item
3 - Ver o topo da pilha
4 - Desempilhar um item
5 - Imprimir a pilha
6 - Destruir a pilha
7 - Sair
Escolha uma opção: 5
Pilha Vazia!
```