

1º Trabalho

Curso: Engenharia da Computação
Disciplina: Estruturas de Dados
Prof. Jarbas Joaci de Mesquita Sá Junior
Universidade Federal do Ceará – UFC/Sobral

Entrega: 20/04/2024 via e-mail para jarbas_joaci@yahoo.com.br

Obs. 1: Não receberei o trabalho após a data mencionada.

- O trabalho é **opcional**, deverá ser feito **individualmente** e valerá, no máximo, **1,0** ponto.
- Preferencialmente fazer o trabalho usando a IDE Dev-C++.
- Enviar **todos** os arquivos do projeto, exceto os executáveis (.exe). Organizar os arquivos nas pastas q1, q2 e q3.
- O uso da diretiva `#include` sem um header file (.h) implicará nota zero no código. Por exemplo, **não** usar `#include "nomearquivo.c"`

1ª) Implemente o Tipo Abstrato de Dados (TAD) “lista.h” (ver slides sobre Lista Encadeada) e acrescente as seguintes funções:

a) função que calcule o número de nós de uma lista. Essa função deve obedecer ao protótipo:

```
int comprimento(Lista* l);
```

b) função para retornar o número de nós da lista que possuem o campo `info` com valor menor que `n`. Essa função deve obedecer ao protótipo:

```
int menores(Lista* l, int n);
```

c) função para somar os valores do campo `info` de todos os nós. Essa função deve obedecer ao protótipo:

```
int soma(Lista* l);
```

d) função para retornar o número de nós da lista que possuem o campo `info` com número **primo**. Essa função deve obedecer ao protótipo:

```
int primos(Lista* l);
```

e) função para criar uma lista que é a concatenação de uma lista `l2` no final de uma lista `l1`. Essa função deve obedecer ao protótipo:

```
List* lst_conc(Lista* l1, Lista* l2);
```

Obs. as listas `l1` e `l2` permanecem inalteradas após a execução da função.

f) função que faça a diferença de duas listas L_1 e L_2 (ou seja, que retire de L_1 os elementos que estão em L_2). Por exemplo, se lista $L_1 \rightarrow 3 \rightarrow 7 \rightarrow 2 \rightarrow 4 \rightarrow //$ e lista L_2

→ 7 → 9 → //, a lista L_1 modificada deve ser $L_1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow //$. Essa função deve obedecer ao protótipo:

```
Lista* lst_diferenca(Lista* l1, Lista* l2);
```

A seguir, execute o seguinte programa.

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

int main(void){
    Lista* l1 = lst_cria();
    l1 = lst_insere(l1,60);
    l1 = lst_insere(l1,13);
    l1 = lst_insere(l1,45);
    l1 = lst_insere(l1,28);
    l1 = lst_insere(l1,41);
    l1 = lst_insere(l1,37);
    l1 = lst_remove_rec(l1,41);
    lst_imprime(l1);
    lst_imprime_invertida_rec(l1);
    printf("Num. nós c/ info < que 30: %d\n",menores(l1,30));
    printf("O comprimento da lista é %d\n",comprimento(l1));
    printf("Soma dos valores dos nós %d\n",soma(l1));
    printf("Num. nós com val. primos é %d\n",primos(l1));

    Lista* l2 = lst_cria();
    l2 = lst_insere(l2,28);
    l2 = lst_insere(l2,45);
    l2 = lst_insere(l2,130);

    Lista* l3=lst_conc(l1,l2);
    lst_imprime(l3);

    l1=lst_diferenca(l1,l2);
    lst_imprime(l1);

    lst_libera(l1);
    lst_libera(l2);
    lst_libera(l3);

    system("PAUSE");
    return 0;
}
```

2ª) Implemente o Tipo Abstrato de Dados (TAD) “pilha.h” usando Vetor (ver slides sobre Pilha) e acrescente as seguintes funções:

a) função que receba uma pilha como argumento e retorne o valor armazenado em seu topo. Essa função deve obedecer ao protótipo:

```
int topo(Pilha* p);
```

Obs. Essa função não altera a pilha, apenas retorna uma cópia do valor armazenado no seu topo.

b) função que retorne o número de elementos da pilha que possuem o campo `info` com valor ímpar. Essa função deve obedecer ao protótipo:

```
int impares(Pilha* p);
```

c) função que verifique quais são os elementos pares de uma pilha `p1` e que os empilhe em **ordem crescente** em uma pilha `p2`. Essa função deve obedecer ao protótipo:

```
Pilha* empilha_pares(Pilha* p1, Pilha* p2);
```

Obs. a pilha `p1` permanece inalterada após a execução da função.

A seguir, execute o seguinte programa.

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

int main(void){
    int a;
    Pilha* p1 = pilha_cria();
    pilha_push(p1,10);
    pilha_push(p1,20);
    pilha_push(p1,25);
    pilha_push(p1,30);
    a = pilha_pop(p1);
    pilha_imprime(p1);
    printf("Elemento no topo da pilha p1: %d\n",topo(p1));
    printf("Qde elems impares na pilha p1: %d\n",impares(p1));

    Pilha* p2 = pilha_cria();
    pilha_push(p2,3);
    pilha_push(p2,4);
    pilha_push(p2,5);
    p2 = empilha_pares(p1,p2);
    pilha_imprime(p2);

    pilha_libera(p1);
    pilha_libera(p2);

    system("PAUSE");
    return 0;
}
```

3ª) Implemente o Tipo Abstrato de Dados (TAD) “fila.h” usando Lista Encadeada (ver slides sobre Fila) e acrescente as seguintes funções:

a) função para retornar o número de elementos da fila com valor maior que n. Essa função deve obedecer ao protótipo:

```
int qtd_maior(Fila* f, int n);
```

b) função que crie uma fila com os elementos da fila f na ordem inversa. Essa função deve obedecer ao protótipo:

```
Fila* inverte(Fila* f);
```

c) função para retornar o número de elementos da fila que possuem o campo info com valor par. Essa função deve obedecer ao protótipo:

```
int pares(Fila* f);
```

A seguir, execute o seguinte programa.

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

int main(void){
    int a, qtd;
    Fila* f1 = fila_cria();
    fila_insere(f1,11);
    fila_insere(f1,12);
    fila_insere(f1,13);
    fila_insere(f1,14);
    fila_insere(f1,15);
    a = fila_remove(f1);
    printf("'Valor removido da fila f1: %d\n',a);
    fila_imprime(f1);

    Fila* f2=inverte(f1);
    fila_imprime(f2);

    qtd=qtd_maior(f1,12);
    printf("'Núm. de elem. maiores que 12 em f1: %d\n',qtd);
    printf("'Qtd. elem. pares na fila f1: %d\n',pares(f1));

    fila_libera(f1);
    fila_libera(f2);

    system("PAUSE");
    return 0;
}
```