

ATIVIDADE 2 - ESOFT - ESTRUTURA DE DADOS I - 52/2022**Período:**23/05/2022 08:00 a 10/06/2022 23:59 (Horário de Brasília)**Status:**ABERTO**Nota máxima:**1,00**Gabarito:**Gabarito será liberado no dia 02/07/2022 00:00 (Horário de Brasília)**Nota obtida:****1ª QUESTÃO**

Considere o excerto de código a seguir:

```
void remover() {  
    if (estr.ini == estr.fim) {  
        printf("\nA estrutura está vazia, não há nada para remover!\n\n");  
        system("pause");  
    }else {  
        int i;  
        for (i = 0; i < tamanho; i++) {  
            estr.dados[i] = estr.dados[i+1];  
        }  
        estr.dados[filas.fim] = 0;  
        estr.fim--;  
    }  
}
```

O algoritmo representa a remoção de elementos de um determinado tipo de estrutura de dados.

Assinale a alternativa que indica o tipo de estrutura que o algoritmo se refere.

ALTERNATIVAS

- ☒ Fila
- ☐ Lista
- ☐ Pilha
- ☐ Heap
- ☐ Grafo

2ª QUESTÃO

O algoritmo a seguir implementa a função imprimir de uma lista simplesmente encadeada. Lembrando que o nó da estrutura de dados de uma lista simplesmente encadeada possui apenas um ponteiro para o próximo elemento da lista.

```
lista ← ptLista
```

```
enquanto lista <> NULL faça
```

```
    imprimir (lista.info)
```

```
    lista ← lista.prox
```

Oliveira, P. M.; Pereira, R. de L. Estrutura de Dados I. Maringá-Pr.: Unicesumar, 2019.

Assinale a alternativa com a correta implementação desse algoritmo em linguagem C:

ALTERNATIVAS

```
void imprimir(){  
    for (int i =0; i < 10;){  
        printf("%d, ", info
```

i

```
    );  
    i++;  
    }  
}
```

```
void imprimir(){  
    lista = ptLista;  
    while(lista != 0) {  
        printf("%d, ", lista->info);  
        lista = lista->prox;  
    }  
}
```

```
void imprimir(){  
    lista = ptLista;  
    while(lista->prox != NULL) {  
        printf("%d, ", lista->info);  
        lista = lista->prox;  
    }  
}
```

```
void imprimir(lista = ptLista){  
    do{  
        printf("%d, ", lista->info);  
        lista = lista->prox;  
    } while(lista->prox != NULL)  
}  
  
void imprimir(info){  
    lista = ptLista;  
    while(lista->info != NULL) {  
        printf("%d, ", lista->prox);  
        lista = lista->info;  
    }  
}
```

3ª QUESTÃO

Observe o excerto de código a seguir:

```
#include <stdio.h>  
  
int main(void) {  
    i = 1;  
    while (i <= 10);  
    ++i;  
}  
}
```

Esse código contém erros. Assinale a alternativa que corresponde a correção desse erro.

ALTERNATIVAS

Esse código estará sempre errado porque não provê uma saída.

```
#include
int main(void) {
    i = 1;
    while (i <= 10){
        ++i;
    }
}
```

```
#include
int main(void) {
    int i = 1;
    while (i <= 10){
        ++i;
    }
}
```

```
#include
int main(void) {
    while (i <= 10){
        int i = 1;
        ++i;
    }
}
```

```
#include
int main(void) {
    int i == 1;
    while (i >= 10){
        ++i;
    }
}
```

4ª QUESTÃO

Na informática, a pilha é uma estrutura em que os dados são inseridos e removidos no seu topo. São estruturas conhecidas como Last In, First Out (LIFO), que pode ser traduzido por último a entrar, primeiro a sair. A operação de entrada lê o dado diretamente na primeira posição disponível, que representa o topo da pilha, essa posição é guardada no atributo fim. Depois da leitura, o valor de fim é atualizado para que ele aponte sempre para a primeira posição disponível. A operação de saída se dá no elemento fim -1 do vetor dados, uma vez que o atributo fim aponta para a primeira posição livre. Após a remoção do item, o valor de fim deve ser atualizado para apontar corretamente para o final da pilha que acabou de diminuir.

Oliveira, P. M.; Pereira, R. de L. Estrutura de Dados I. Maringá-Pr.: Unicesumar, 2019.

Com relação à estrutura de pilha e às operações associadas a esse tipo de estrutura, assinale a alternativa que apresenta os nomes dos métodos corretamente segundo o referencial bibliográfico da disciplina.

ALTERNATIVAS

- ☒ Push(x) e Pop() são os métodos para inserção e remoção, respectivamente.
- ☐ Root(x) e Front() são os métodos para inserção e remoção, respectivamente.
- ☐ SetLast(x) e GetFront() são os métodos para inserção e remoção, respectivamente.
- ☐ Dequeue(x) e Enqueue são os métodos para inserção e remoção, respectivamente.
- ☐ AddFirst(x) e DelLast() SetLast() e GetFront() são os métodos para inserção e remoção, respectivamente.

5ª QUESTÃO

Você faz parte de uma equipe de programadores contratados para implementar uma solução de grafo ponderado representado por uma lista de adjacências.

Primeiro foi criada a lista de adjacências, que aponta para o nó de destino, o peso associado a aresta que leva ao nó de destino e o próximo elemento da lista de adjacências.

```
typedef struct adjacencia {  
    int vertice;  
    int peso;  
    struct adjacencia *prox;  
}ADJACENCIA;
```

Os dados são armazenados nos vértices ou nós. Esta estrutura possui apenas a cabeça da lista de adjacências.

```
typedef struct vertice {  
    ADJACENCIA *cabeca;  
}VERTICE;
```

A estrutura do grafo armazena o número total de vértices e arestas do grafo caso você queira utilizar em algum momento do seu código. A vantagem em saber desta informação consiste em evitar de percorrer toda a estrutura. A última linha de definição da estrutura contém o arranjo de vértices da estrutura.

```
typedef struct grafo {  
    int vertices;  
    int arestas;  
    VERTICE *adj;  
}GRAFO;
```

Após a definição das estruturas (lista de adjacência, vértice e grafo), os desenvolvedores implementaram a função de criar o grafo.

```
GRAFO *criaGrafo (int v) {  
    int i;  
    GRAFO *g = (GRAFO *)malloc(sizeof(GRAFO));  
    g->vertices = v;  
    g->arestas = 0;  
    g->adj = (VERTICE *)malloc(v*sizeof(VERTICE));  
    for (i=0; i<v; i++){  
        g->adj[i].cabeca = NULL;  
    }  
    return(g);  
}
```

Além de implementar o código, assinale a alternativa correta a respeito da função malloc a seguir:

```
GRAFO *g = (GRAFO *)malloc(sizeof(GRAFO));
```

ALTERNATIVAS

- ☐ A função malloc retorna o tamanho da estrutura do grafo.
- ☐ A função malloc reservará um espaço na memória relativo a 8 bits.
- ☐ A função malloc reservará um espaço na memória relativo a 16 bytes.
- ☒ A função malloc reservará um espaço na memória do tamanho de um inteiro.
- ☐ A função malloc reservará em memória um espaço relativo a estrutura do grafo.

