Ciência da Computação
Estrutura de Dados I
Prof. André Kishimoto

Observação: As instruções sobre entrega, prazo e avaliação estão nas páginas 3-4 do documento.

Lista de Exercícios 4 (Lista ligada/encadeada) – EM DUPLA

Baseado no TAD Lista Ligada/Encadeada e pseudocódigo apresentado em aula (veja slides da semana 10), escreva sua própria implementação da lista ligada/encadeada em linguagem C++.

O seu arquivo main.cpp deve conter exatamente o código abaixo, com exceção das linhas com comentários TODO, que devem ser substituídos pelas instruções corretas da sua implementação da lista.

Atualização 27/04/2021:

- A função Print(const LinkedList* list); que é chamada dentro da função PrintListInfo() abaixo deve ser implementada pela dupla.
- Prazo de entrega alterado de até 02/05/2021 23:59 para até 05/05/2021 23:59.

```
// main.cpp
#include <iostream>
#include <clocale>
#include "LinkedList.h"

using namespace std;

void Print(const LinkedList* list)
{
    // TODO: Implementar:
    // Percorre todos os nós da lista e imprime os valores de cada nó.
}

void PrintListInfo(const LinkedList* list)
{
    if (IsEmpty(list))
    {
        cout << "Lista vazia!\n";
    }
    else
    {
        cout << "Lista: ";
        Print(list);
    }
}

int main()
{
    setlocale(LC_ALL, "pt_BR");
    cout << "*** Lista Ligada/Encadeada (Linked List) ***\n";

    LinkedList* list = Create();
```

```
    PrintListInfo(list);

    Insert(list, 1);
    Insert(list, 2);
    Insert(list, 3);
    Append(list, 4);
    Append(list, 5);
    Append(list, 6);
    PrintListInfo(list);

    Clear(list);
    PrintListInfo(list);

    Insert(list, 77);
    Append(list, 88);
    Insert(list, 99);
    Append(list, 3);
    Insert(list, 2);
    Append(list, 1);
    Insert(list, 0);
    PrintListInfo(list);

    Node* temp = RemoveNode(list, 3);
    cout << "Nó removido: " << temp->data << '\n';
    // TODO: Liberar memória alocada para o nó que foi removido.
    PrintListInfo(list);

    temp = RemoveHead(list);
    cout << "Nó removido: " << temp->data << '\n';
    // TODO: Liberar memória alocada para o nó que foi removido.
    PrintListInfo(list);

    temp = RemoveTail(list);
    cout << "Nó removido: " << temp->data << '\n';
    // TODO: Liberar memória alocada para o nó que foi removido.
    PrintListInfo(list);

    // TODO: Liberar memória alocada para a lista.

    cout << "Fim.\n";
}
```

A execução do código acima reproduz a seguinte saída:

```
*** Lista Ligada/Encadeada (Linked List) ***
Lista vazia!
Lista: 3 2 1 4 5 6
Lista vazia!
Lista: 0 2 99 77 88 3 1
Nó removido: 3
Lista: 0 2 99 77 88 1
Nó removido: 0
Lista: 2 99 77 88 1
Nó removido: 1
Lista: 2 99 77 88
Fim.
```

Desenvolvimento (10,0 pontos)

- Sua solução deve ser escrita apenas com a linguagem C++.
- A sua implementação da lista ligada/encadeada deve satisfazer as instruções da main.cpp listada nesse documento e a execução do código deve reproduzir a mesma saída indicada na página anterior.

Identificação e referências

- Coloque sua identificação - nome e TIA - no início de cada arquivo de código, como comentário (use // no começo de cada linha que queira comentar).
- Inclua como comentário quaisquer referências (livros, artigos, sites, entre outros) usadas para solucionar o problema.

Entrega

- **Código:** Compacte todos os arquivos .cpp/.h ou o projeto completo criado na IDE que você está usando (mas sem os intermediários como bin e obj) no formato zip OU comite todos os arquivos .cpp/.h ou o projeto completo criado na IDE que você está usando (mas sem os intermediários como bin e obj) em um repositório git.
- **Arquivo texto (.txt):**
  - Se o código está em um repositório git, envie um arquivo txt no Moodle contendo sua identificação e o link do repositório.
- **Prazo de entrega:** via link do Moodle até 05/05/2021 23:59.

Informações importantes sobre critérios de avaliação

Embora essa atividade seja uma avaliação da disciplina, sempre considero que as atividades também podem ser usadas para nos acostumarmos com o mercado de trabalho. Portanto, leve em consideração os seguintes critérios que vou aplicar na avaliação:

- Será descontado 1,0 (um) ponto caso a entrega não respeite o enunciado. Exemplos:
  - O enunciado pede para enviar um arquivo compactado no formato zip, mas o arquivo enviado está no formato rar.
  - O enunciado pede um arquivo texto no formato txt, mas foi enviado um documento do Word.
  - Não há identificação nem referências (caso aplicável) nos arquivos de código.

- Será descontado 1,0 (um) ponto caso o arquivo zip OU o repositório git contenha pastas e arquivos desnecessários.
  Exemplo:
  - Pastas intermediárias criadas no processo de compilação (Debug, obj, bin, ...).

- O projeto deve ser desenvolvido em linguagem C++ e não em linguagem C. Caso a solução apresentada use funcionalidades da linguagem C e que tenham equivalentes em C++, será descontado 2,0 (dois) pontos.
  Atente-se a esse detalhe quando estiver pesquisando e verificando exemplos na internet e outros materiais, principalmente de assuntos que não vimos até o momento (essa atividade pode ser resolvida só com o que foi visto em aula, com suas devidas adaptações).

Exemplo:

- Declarar arrays de tamanho variável (padronizado no C99, mas erro em C++ pois não há suporte para VLA), ex. int n = 10; char arr[n];.

- Projeto que possui erros de compilação ou que trava durante a execução automaticamente perde 50% da nota máxima.
  Sobre erros de compilação: considero apenas erros, não há problema se o projeto tiver warnings (apesar que warnings podem avisar possíveis travamentos em tempo de execução, como loop infinito, divisão por zero etc.).
  Quando há necessidade de entrada de dados por parte do usuário, assumo que o usuário vai inserir as informações corretas (ex. tipos de dados corretos), a menos que o enunciado explicite que você deve garantir que os dados de entrada estejam corretos.

Em uma situação profissional, os itens indicados acima atrapalham (e muito) o trabalho da equipe. E o último item é gravíssimo (o ideal também é remover todos os warnings e sempre validar os dados).