

**Alunos: Deivison Correia Lima e Karolyne Imaculada Andrade Muniz**

**Curso: Ciência da Computação - DACC/UNIR**

**Trabalho 1: Listas Lineares**

**Estrutura de Dados I-- 2022 -- DACC/UNIR, Profa. Carolina Watanabe**

- **Sobre o trabalho**

O trabalho consiste na criação de uma lista de alunos, estática e dinâmica, para armazenamento de dados que consistem na matrícula, nome, nota da primeira, segunda e terceira avaliações. Ambas implementações salvam os dados que foram inseridos pelo usuário em um arquivo .txt que se encontra na pasta do projeto.

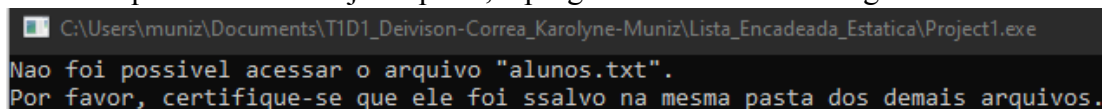
A matrícula é um tipo de dado que não pode exceder seis dígitos e o nome não pode exceder trinta caracteres.

- **Estrutura de dados**

Ambos projetos apresentam estrutura de dados diferentes. A lista estática apresenta um vetor MAX 101. A lista dinâmica não apresenta vetor e utiliza ponteiros para a alocação dos dados(head).

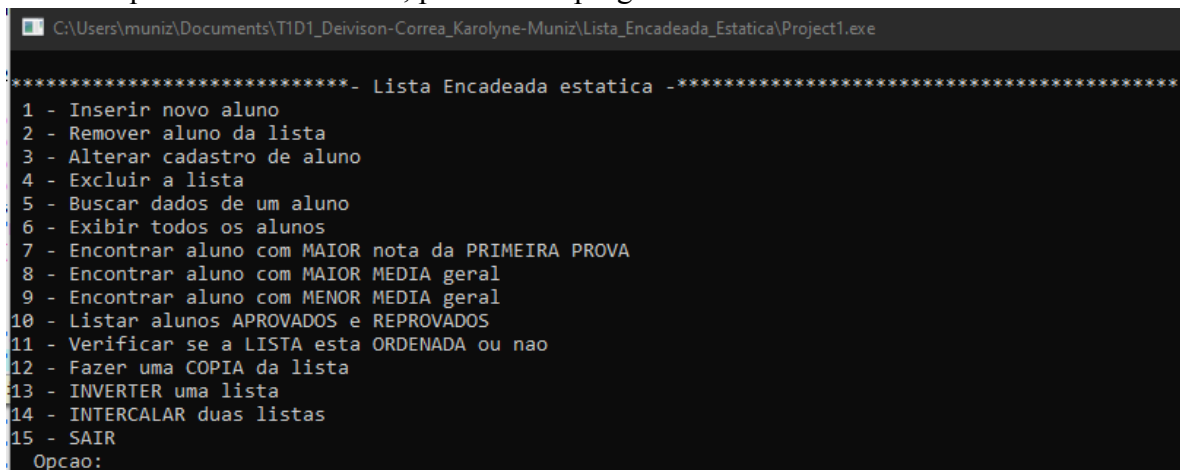
- **Testes**

Caso o arquivo txt não esteja na pasta, o programa abre uma mensagem de erro.



```
C:\Users\muniz\Documents\T1D1_Deivison-Correa_Karolyne-Muniz\Lista_Encadeada_Estatica\Project1.exe
Nao foi possivel acessar o arquivo "alunos.txt".
Por favor, certifique-se que ele foi ssalvo na mesma pasta dos demais arquivos.
```

Com o arquivo no local correto, pode usar o programa normalmente.



```
C:\Users\muniz\Documents\T1D1_Deivison-Correa_Karolyne-Muniz\Lista_Encadeada_Estatica\Project1.exe
*****- Lista Encadeada estatica -*****
1 - Inserir novo aluno
2 - Remover aluno da lista
3 - Alterar cadastro de aluno
4 - Excluir a lista
5 - Buscar dados de um aluno
6 - Exibir todos os alunos
7 - Encontrar aluno com MAIOR nota da PRIMEIRA PROVA
8 - Encontrar aluno com MAIOR MEDIA geral
9 - Encontrar aluno com MENOR MEDIA geral
10 - Listar alunos APROVADOS e REPROVADOS
11 - Verificar se a LISTA esta ORDENADA ou nao
12 - Fazer uma COPIA da lista
13 - INVERTER uma lista
14 - INTERCALAR duas listas
15 - SAIR
Opcao:
```

Programa exibindo os dados.

```
C:\Users\muniz\Documents\T1D1_Deivison-Correa_Karolyne-Muniz\Lista_Encadeada_Estatica\Project1.exe

EXIBIR TODOS OS ALUNOS

Lista Principal
101001 - Ana Silva 8.90 7.80 6.53 - 0.00, 0.00, 0.00
101103 - Anastacia Lima 7.25 8.97 6.4 - 0.00, 0.00, 0.00
101203 - Fernanda Ascencio 8.59 9.53 6.82 - 0.00, 0.00, 0.00
101304 - Luiza Araujo 5.2 5.1 3.7 - 0.00, 0.00, 0.00

Lista Copiada
Nao ha elementos na lista

Lista Invertida
Nao ha elementos na lista

Lista Intercalada
Nao ha elementos na lista

Aperte uma tecla para retornar ao menu
```

Inserção de um novo aluno.

```
C:\Users\muniz\Documents\T1D1_Deivison-Correa_Karolyne-Muniz\Lista_Encadeada_Estatica\Project1.exe

CADASTRAR ALUNO (na lista principal)

Deseja inserir o novo aluno em qual posicao:
1 - Inicio da Lista
2 - Final da Lista
3 - Inserir ordenando (automaticamente) pela matricula
Opcao: 3

Digite a matricula: 101209

Digite o nome do aluno: Gracielly Ramos

Digite a nota 1: 9.7

Digite a nota 2: 7.4

Digite a nota 3: 5.9

Aluno inserido ORDENADAMENTE com sucesso

Aperte uma tecla para retornar ao menu
```

Exclusão de um aluno.

```
C:\Users\muniz\Documents\T1D1_Deivison-Correa_Karolyne-Muniz\Lista_Encadeada_Estatica\Project1.exe

EXIBIR TODOS OS ALUNOS

Lista Principal
101001 - Ana Silva 8.90 7.80 6.53 - 0.00, 0.00, 0.00
101103 - Anastacia Lima 7.25 8.97 6.4 - 0.00, 0.00, 0.00
101203 - Fernanda Ascencio 8.59 9.53 6.82 - 0.00, 0.00, 0.00
101209 - Gracielly Ramos - 9.80, 7.40, 5.90
101304 - Luiza Araujo 5.2 5.1 3.7 - 0.00, 0.00, 0.00

Lista Copiada
Nao ha elementos na lista

Lista Invertida
Nao ha elementos na lista

Lista Intercalada
Nao ha elementos na lista

Aperte uma tecla para retornar ao menu
```

```
C:\Users\muniz\Documents\T1D1_Deivison-Correa_Karolyne-Muniz\Lista_Encadeada_Estatica\Project1.exe

EXIBIR TODOS OS ALUNOS

Lista Principal
101001 - Ana Silva 8.90 7.80 6.53 - 0.00, 0.00, 0.00
101103 - Anastacia Lima 7.25 8.97 6.4 - 0.00, 0.00, 0.00
101203 - Fernanda Ascencio 8.59 9.53 6.82 - 0.00, 0.00, 0.00
101304 - Luiza Araujo 5.2 5.1 3.7 - 0.00, 0.00, 0.00

Lista Copiada
Nao ha elementos na lista

Lista Invertida
Nao ha elementos na lista

Lista Intercalada
Nao ha elementos na lista

Aperte uma tecla para retornar ao menu
```

Alterar dados de um aluno.

```
C:\Users\muniz\Documents\T1D1_Deivison-Correa_Karolyne-Muniz\Lista_Encadeada_Estatica\Project1.exe

ALTERAR DADOS DE ALUNO (na lista principal)
Digite a matricula do aluno a ter seus dados alterados: 101001

Deseja alterar a matricula do aluno? [S/N]: n

Deseja alterar o nome do aluno? [S/N]: s
Digite o novo nome do aluno: Ana Silva Ramos

Deseja alterar a nota 1 do aluno? [S/N]: n

Deseja alterar a nota 2 do aluno? [S/N]: n

Deseja alterar a nota 3 do aluno? [S/N]: n

Cadastro alterado com sucesso
```

```
C:\Users\muniz\Documents\T1D1_Deivison-Correa_Karolyne-Muniz\Lista_Encadeada_Estatica\Project1.exe

EXIBIR TODOS OS ALUNOS

Lista Principal
101001 - Ana Silva Ramos - 0.00, 0.00, 0.00
101103 - Anastacia Lima 7.25 8.97 6.4 - 0.00, 0.00, 0.00
101203 - Fernanda Ascencio 8.59 9.53 6.82 - 0.00, 0.00, 0.00
101304 - Luiza Araujo 5.2 5.1 3.7 - 0.00, 0.00, 0.00
```

Maior nota na primeira prova.

```
C:\Users\muniz\Documents\TID1_Deivison-Correa_Karolyne-Muniz\Lista_Encadeada_Estatica\Project1.exe

MAIOR NOTA:
- (111111) Ana Paula tirou 8.60 na primeira prova

Aperte uma tecla para retornar ao menu
```

### Menor nota

```
C:\Users\muniz\Documents\TID1_Deivison-Correa_Karolyne-Muniz\Lista_Encadeada_Estatica\Project1.exe

MENOR MEDIA:
- (222222) Fernanda Dias teve media 6.33

Aperte uma tecla para retornar ao menu
```

### Maior media(Luiza)

```
C:\Users\muniz\Documents\TID1_Deivison-Correa_Karolyne-Muniz\Lista_Encadeada_Estatica\Project1.exe

MAIOR MEDIA:
- (333333) $-+ teve media 7.87

Aperte uma tecla para retornar ao menu
```

### Alunos aprovados e reprovados.

```
C:\Users\muniz\Documents\TID1_Deivison-Correa_Karolyne-Muniz\Lista_Encadeada_Estatica\Project1.exe

ALUNOS APROVADOS:
- (444444) Ophelia Guimaraes teve media: 6.53
- (333333) Luiza Lima teve media: 7.87
- (222222) Fernanda Dias teve media: 6.33
- (111111) Ana Paula teve media: 7.20

ALUNOS REPROVADOS:

Aperte uma tecla para retornar ao menu
```

## ● Operações

1. int Gravar\_Dados(Lista \*L)/ int Ler\_Dados(Lista \*L);

Pré-condição: L é uma lista que existe (mesmo vazia). Pós-condição: retorna 1 se os dados de um arquivo texto foram gravados/carregados para L, retorna 0 se isso não ocorreu.

2. int Ler\_Dados(Lista \*L);

Pré-condição: L é uma lista que existe (mesmo vazia). Pós-condição: retorna 1 se os dados foram gravados num arquivo texto, retorna 0 se isso não ocorreu.

3. void Iniciar(Lista \*L);

Pré-condição: L é uma variável do tipo lista. Pós-condição: faz o início de L apontar para NULL.

4. int Vazia(Lista \*L);

Pré-condição: L é uma lista que existe. Pós-condição: retorna 1 se o início de L aponta para NULL (isto é, se L está vazia), retorna 0 no caso contrário.

5. int Verificar\_Chave(Lista \*L, int chave);

Pré-condição: L é uma lista que existe (mesmo vazia), chave contém um valor inteiro. Pós-condição: retorna 1 se a chave informada já existe em L; retorna 0 se não existe.

6. int Inserir\_Aluno\_Inicio(Lista \*L, tipo\_elem v);

Pré-condição: L é uma lista que existe (mesmo vazia), v possui valores próprios.  
Pós-condição: retorna 1 se um novo nó pôde ser alocado, se os dados de v foram inseridos nesse nó e se esse nó foi encadeado ao início de L; retorna 0 se isso não ocorreu.

7. int Inserir\_Aluno\_Final(Lista \*L, tipo\_elem v);

Pré-condição: L é uma lista que existe (mesmo vazia), v possui valores próprios.  
Pós-condição: depois de recorrer a outras funções de inserção, retorna 1 se um novo nó pôde ser alocado, se os dados de v foram inseridos nesse nó e se esse nó foi encadeado posteriormente à última posição de L; retorna 0 se isso não ocorreu.

8. int Inserir\_Aluno\_Ordenado(Lista \*L, tipo\_elem v);

Pré-condição: L é uma lista que existe (mesmo vazia), v possui valores próprios.  
Pós-condição: depois de recorrer a outras funções de inserção, retorna 1 se um novo nó pôde ser alocado, se os dados de v foram inseridos nesse nó e se esse nó foi encadeado ordenadamente (de maneira crescente ou decrescente) à L, retorna 0 se isso não ocorreu.

9. int Inserir\_Aluno\_Apos(Lista \*L, tipo\_elem v, No \*k);

Pré-condição: L é uma lista que existe (mesmo vazia), v possui valores próprios.  
Pós-condição: retorna 1 se um novo nó pôde ser alocado, se os dados de v foram inseridos nesse nó e se esse nó foi encadeado após o nó k de L; retorna 0 se isso não ocorreu.

10. int Remover\_Aluno(Lista \*L, int chave);

Pré-condição: L é uma lista que existe (mesmo vazia), chave possui um valor inteiro.  
Pós-condição: o aluno com a matrícula passada como chave é retirado da lista (e seu registro no arquivo texto é removido); retorna 1 se a remoção foi bem sucedida, retorna 0 se isso não ocorreu. Caso a lista esteja vazia, o usuário é informado disso.

11. int Alterar\_Cadastro(Lista \*L, tipo\_elem v, int k);

Pré-condição: L é uma lista que existe (mesmo vazia), v possui valores próprios, k possui um valor inteiro. Pós-condição: o aluno com a matrícula passada como k tem seus dados alterados pelos novos dados recebidos de “v”, a não ser que os valores de “v” não se enquadrem em valores plausíveis (por exemplo, se a nota 2 for -1, não há alteração real dessa nota, pois -1 não é plausível) é retirado da lista (e seu registro no arquivo texto é removido); retorna 1 se a remoção foi bem sucedida, retorna 0 se isso não ocorreu. Caso a lista esteja vazia, o usuário é informado disso.

12. void Excluir\_Lista(Lista \*L);

Pré-condição: L é uma lista que existe (mesmo vazia). Pós-condição: todos os nós de L recebem NULL, por fim seu início também recebe NULL. Nesse sentido, a lista se torna vazia. Uma lista já vazia passada como parâmetro continua vazia.

13. int Buscar\_Aluno(Lista \*L, int chave);

Pré-condição: L é uma lista que existe (mesmo vazia), k é um número inteiro plausível para uma matrícula (números negativos, por exemplo, não são plausíveis). Pós-condição: caso a chave corresponda de fato a alguma matrícula na lista, a própria função Buscar imprime na tela os dados. Se não há correspondência, uma mensagem de aviso é exibida; retorna 1 se encontrou o aluno da matrícula, retorna 0 no caso contrário. Caso a lista esteja vazia, o usuário é informado disso.

14. void Exibir\_Tudo(Lista \*L);

Pré-condição: L é uma lista que existe (mesmo vazia). Pós-condição: todos os dados, de todos os alunos, de todas as listas são impressos na tela do usuário. Caso a lista esteja vazia, o usuário é informado disso.

15. void Maior\_Nota1(Lista \*L);

Pré-condição: L é uma lista que existe (mesmo vazia). Pós-condição: encontrada a maior nota da primeira prova, todos os alunos com esta nota são exibidos - impressos pela própria função. Caso a lista esteja vazia, o usuário é informado disso.

16. void Maior\_Media(Lista \*L);

Pré-condição: L é uma lista que existe (mesmo vazia). Pós-condição: encontrada a maior média, todos os alunos com esta nota são exibidos - impressos pela própria função. Caso a lista esteja vazia, o usuário é informado disso.

17. void Menor\_Media(Lista \*L);

Pré-condição: L é uma lista que existe (mesmo vazia). Pós-condição: encontrada a menor média, todos os alunos com esta nota são exibidos - impressos pela própria função. Caso a lista esteja vazia, o usuário é informado disso.

18. void Listar\_Situacao(Lista \*L);

Pré-condição: L é uma lista que existe (mesmo vazia). Pós-condição: ocorre a impressão de uma lista com alunos aprovados e de uma com os reprovados. Caso a lista esteja vazia, o usuário é informado disso.

19. int Verificar\_Ordem(Lista \*L);

Pré-condição: L é uma lista que existe (mesmo vazia). Pós-condição: Quatro casos são abordados: ordem crescente (retorna 2), ordem decrescente (retorna 1), se a lista tem apenas um elemento, caso que não é possível determinar ordem (retorna 3) e, por fim, lista desordenada (retorna 0). Caso a lista esteja vazia, o usuário é informado disso.

20. int Copia\_Lista(Lista \*L, Lista \*L2);

Pré-condição: L e L2 são listas que existem. Pós-condição: todo o conteúdo de L é copiado para L2, seguindo a distribuição dos alunos na lista L. Caso L esteja vazia, a operação não pode ser realizada, e o usuário é informado disso; se L2 já tiver conteúdo, este é perdido para a nova cópia.

21. int Inverter\_Lista(Lista \*L, Lista \*L2);

Pré-condição: L e L2 são listas que existem. Pós-condição: L2 recebe os conteúdos de L na ordem inversa da qual estão em L. Caso L esteja vazia, a operação não pode ser realizada, e o usuário é informado disso.

22. int Intercalar\_Listas(Lista \*L1, Lista \*L2, Lista \*L3);

Pré-condição: L1, L2 e L3 são listas que existem, L1 e L2 estão ordenadas. Pós-condição: L3 recebe os conteúdos de L1 e L2. Se L1 e L2 são crescente, L3 estará em ordem crescente; se L1 e L2 são decrescentes, L3 também o será; se L1 e L2 são ordenadas, mas de maneira inversa, L3 será intercalada obedecendo à ordem crescente. Caso pelo menos uma das listas esteja vazia, a operação não pode ser realizada, e o usuário é informado disso.