Instituto Federal do Amazonas
Curso Superior em Análise e Desenvolvimento de Sistemas
Professor: Sergio Augusto C. Bezerra sergio.bezerra@ifam.edu.br
Disciplina: Estrutura de Dados

LISTAS LINEARES

- Definição
- Aplicabilidade
- Listas Encadeadas
- Tipos de Listas Encadeadas
  - Encadeamento Simples
  - Duplamente Encadeadas
  - Ordenadas
  - Circulares

1) Alocação de memória.

```
/* programa "alocmem1.c" para exemplificar a alocação de memória */
#include <stdio.h>
main(){

  int x, *p, *q;

  p = (int *) malloc(sizeof (int));
  *p = 5;
  q = p;
  printf("*p = %d  *q = %d\n", *p, *q);
  x = 10;
  *q = x;
  printf("*p = %d  *q = %d\n", *p, *q);
  p = (int *) malloc(sizeof (int));
  *p = 8;
  printf("*p = %d  *q = %d\n", *p, *q);
  getchar();
}
```

2) Liberando memória.

```
/* programa "alocmem2.c" para exemplificar a liberação de memória */
#include <stdio.h>
main(){

  int *p, *q;
```

```
  p = (int *) malloc(sizeof (int));
  *p = 20;
  q = (int *) malloc(sizeof (int));
  *q = 30;
  free(p);
  q = (int *) malloc(sizeof (int));
  *q = 50;
  printf("*p = %d  *q = %d\n", *p, *q);

  getchar();
}
```

3) Programa para armazenar e imprimir notas como uma Lista Simplesmente Encadeada.

```
/*  Programa "LS_nota.cpp" para exemplificar uma lista simplesmente encadeada
    para guardar uma sequência de notas */
#include <stdio.h>
#include <cstdlib>
#include <stdlib.h>
#include <iostream>
using namespace std;

int main(){
  char resp;
  float valor;

  struct ListaNota {
    float nota;  /* valor da nota */
    struct ListaNota *prox_nota; /* ponteiro para próxima nota */
  } inicio, *no;

  system("cls");;
  printf("PROGRAMA PARA GUARDAR NOTAS EM UMA LISTA SIMPLESMENTE ENCADEADA\n");

  inicio.prox_nota = NULL; /* lista vazia */
  no = &inicio; /* aponta para o inicio da lista */

  do{
    no->prox_nota = (struct ListaNota *) malloc(sizeof(struct ListaNota));
    no = no->prox_nota;
    printf("\nDigite a nota: ");
    scanf("%f", &valor);
    no->nota = valor;
    no->prox_nota = NULL;

    printf("Deseja continuar? Sim[S] Nao[outra tecla]---->");
    cin >> resp;
  }while (resp == 'S'||resp == 's');

  /* Exibindo as notas da lista */
  no = inicio.prox_nota;

  printf("\nLISTA DAS NOTAS: ");
  while(no){
    printf("%.1f\t", no->nota);
    no = no->prox_nota;
  }

  printf("\n");
  system("pause");
```

```
  return 0;
}
```

Exercício 1: Refaça o programa anterior para que ele também armazene uma matrícula para cada aluno.

4) Programa para inserir a matricula, o nome e as notas em uma Lista Simplesmente Encadeada.

```
/* Prof.: Sergio Augusto C. Bezerra
   Programa "LS_INS01.CPP" para exemplificar uma lista simplesmente encadeada
   para guardar a matricula e o nome dos alunos com duas notas de uma
   determinada disciplina, alem do calculo da media.
*/

#include <stdio.h>
#include <alloc.h>
#include <conio.h>
#include <ctype.h>
#include <iostream.h>


/****************** VARIAVEIS GLOBAIS ******************/

  char resp;
  int linha,col;

  struct Aluno {
    int matricula;                  // números de 1 a no máximo 40
    char nome[50];
    float nota1, nota2, media;      // valor da nota
    struct Aluno *pProx;            / ponteiro para próxima nota
  } inicio, *pAux;

/****************** FUNCAO PRINCIPAL ******************/


void main(void){

/*** Inicio da Lista ***/

  inicio.pProx = NULL;      // lista vazia
  pAux = &inicio;           // aponta para o inicio da lista

  do{

    clrscr();
    pAux->pProx = new(Aluno);
    pAux = pAux->pProx;
    gotoxy(3,5);
```

```
    cout << "PROGRAMA PARA GERENCIAR A MATRICULA, O NOME E AS NOTAS DE ";
    cout << "\nUM ALUNO USANDO UMA LISTA SIMPLESMENTE ENCADEADA\n";
    cout << "\nMatricula: ";
    cin >> pAux->matricula;
    cout << "\nNome do Aluno: ";
    gets(pAux->nome);
    cout << "\nNota1: ";
    cin >> pAux->nota1;
    cout << "\nNota2: ";
    cin >> pAux->nota2;
    pAux->media = (pAux->nota1 + pAux->nota2)/2;

    pAux->pProx = NULL;

    cout << "Deseja continuar? Sim[S] Não [outra tecla]---->";
    resp = toupper(getch());
   }while (resp == 'S');

   /* Exibindo as notas da lista */
    clrscr();
    pAux = inicio.pProx;
    gotoxy(1,10);
    cout << "Matricula";
    gotoxy(15,10);
    cout << "Nome";
    gotoxy(50,10);
    cout << "Nota1";
    gotoxy(60,10);
    cout << "Nota2";
    gotoxy(70,10);
    cout << "Media";
    linha=12;
    while(pAux){
      gotoxy(1,linha);
      cout << pAux->matricula;
      gotoxy(15,linha);
      cout << pAux->nome;
      gotoxy(50,linha);
      cout << pAux->nota1;
      gotoxy(60,linha);
      cout << pAux->nota2;
      gotoxy(70,linha);
      cout << pAux->media;
      pAux = pAux->pProx;
      linha++;
     }

   getchar();
   printf("\nPressione uma tecla para terminar >>>>");
   getchar();
}
```

5) Uma versão modularizada do programa anterior para exibir, inserir e remover a matricula, o nome e as notas em uma Lista Simplesmente Encadeada.

```
/* Prof.: Sergio Augusto C. Bezerra

   Programa MODULADO "LS_INS02.CPP" para exemplificar uma lista simplesmente encadeada
   para guardar a matricula e o nome dos alunos com duas notas de uma
   determinada disciplina, além do cálculo da média.
*/

#include <stdio.h>
#include <alloc.h>
#include <conio.h>
#include <ctype.h>
#include <iostream.h>


/****************** VARIAVEIS GLOBAIS ******************/

   char professor[50],disciplina[50],turma[10];
   int opcao;
   char resp;
   int linha, col, matTemp;

   struct Aluno {
     int matricula;                 /* numeros de 1 a no maximo 40 */
     char nome[50];
     float notas[3];                /* valores das notas */
     struct Aluno *pProx;           /* ponteiro para o proximo aluno */
   };

   Aluno inicio, *pAux, *pAnt;


/****************** FUNCAO CABECALHO ******************/
void cabecalho(){
   clrscr();
   gotoxy(3,5);
   cout << "PROGRAMA PARA GERENCIAR A MATRICULA, O NOME E AS NOTAS";
   gotoxy(3,7);
   cout << "DE UM ALUNO USANDO UMA LISTA SIMPLESMENTE ENCADEADA\n";
}

/****************** FUNCAO MENU ******************/
void menu(){
   col=15;
   gotoxy(col,10);
   cout << "*************** Menu ***************";
   gotoxy(col,11);
   cout << "*       Exibir.....[1]          *";
   gotoxy(col,12);
   cout << "*       Inserir.....[2]         *";
```

```
  gotoxy(col,13);
  cout << "*        Remover.....[3]          *";
  gotoxy(col,14);
  cout << "*        Sair........[0]          *";
  gotoxy(col,15);
  cout << "*        Digite a opcao:          *";
  gotoxy(col,16);
  cout << "*****************************************";
  gotoxy(42,15);
  cin >> opcao;
}

/******************** FUNCAO DIARIO ********************/
void diario(){
  gotoxy(3,9);
  cout << "PREENCHER OS DADOS DO CABECALHO DO DIARIO";
  gotoxy(3,11);
  cout << "Professor:";
  gets(professor);
  gotoxy(3,13);
  cout << "Disciplina: ";
  gets(disciplina);
  gotoxy(3,15);
  cout << "Turma: ";
  gets(turma);

}


/********************** FUNCAO EXIBIR ********************/
void exibir(){

    pAux = inicio.pProx; /* aponta para o inicio da lista */
    clrscr();
    gotoxy(1,5);
    cout << "*********************************  DIARIO *********************************";
    gotoxy(1,7);
    cout << "Professor: " << professor;
    gotoxy(1,8);
    cout << "Disciplina: " << disciplina;
    gotoxy(1,9);
    cout << "Turma: " << turma;
    linha=11;
    cout << "\n-------------------------------------------------------------------------";
    gotoxy(1,linha);
    cout << "Matricula";
    gotoxy(15,linha);
    cout << "Nome";
    gotoxy(50,linha);
    cout << "Nota1";
    gotoxy(60,linha);
```

```
     cout << "Nota2";
     gotoxy(70,linha);
     cout << "Media";
     linha=13;
     cout << "\n--------------------------------------------------------------------------------";
     while(pAux){
       gotoxy(1,linha);
       cout << pAux->matricula;
       gotoxy(15,linha);
       cout << pAux->nome;
       gotoxy(50,linha);
       cout <<  pAux->notas[0];
       gotoxy(60,linha);
       cout << pAux->notas[1];
       gotoxy(70,linha);
       cout << pAux->notas[2];
       pAux = pAux->pProx;
       linha++;
     }
    cout << "\n--------------------------------------------------------------------------------";
    printf("\nPressione Enter para continuar!");
     getch();
}
/********************** FUNCAO INSERIR ********************/
void inserir(){
   pAux = &inicio; /* aponta para o inicio da lista */
   while(pAux->pProx)
     pAux = pAux->pProx;

   do{
     clrscr();
     cout << "************************** CADASTRO DE ALUNO **************************";
     pAux->pProx = new(Aluno);
     pAux = pAux->pProx;
     gotoxy(1,2);
     cout << "*  Matricula:                                          *";
     gotoxy(1,3);
     cout << "*  Nome do Aluno:                                      *";
     gotoxy(1,4);
     cout << "*  Nota1:                                              *";
     gotoxy(1,5);
     cout << "*  Nota2:                                              *";
     cout << "\n********************************************************************************";
     gotoxy(20,2);
     cin >> pAux->matricula;
     gotoxy(20,3);
     gets(pAux->nome);
     gotoxy(20,4);
     cin >> pAux->notas[0];
     gotoxy(20,5);
     cin >> pAux->notas[1];
```

```
     pAux->notas[2] = (pAux->notas[0] + pAux->notas[1])/2;
     pAux->pProx = NULL;
     cout << "\nContinuar inserindo dados? Sim[S] Nao[outra tecla]---->";
     resp = toupper(getch());
   }while (resp == 'S');
}
/********************** FUNCAO REMOVER ********************/
void remover(){

  resp = '0';
  clrscr();
  cout << "*************************** REMOVER ALUNO ****************************";
  gotoxy(1,2);
  cout << "*  Matricula:                                          *";
  gotoxy(1,3);
  cout << "************************************************************************";
  gotoxy(15,2);
  cin >> matTemp;
  pAux = &inicio;
  while(pAux->matricula!=matTemp && pAux->pProx!=NULL){
    pAnt=pAux;
    pAux = pAux->pProx;
  }
  if(pAux->matricula==matTemp){
    gotoxy(20,2);
    cout << "Remover " << pAux->nome << "? Sim[S] Nao[outra tecla]---->";
    resp = toupper(getch());
    if(resp=='S'){
      pAnt->pProx = pAux->pProx;
      pAux->pProx = NULL;
      pAnt = NULL;
      delete(pAux);
    }
  }
  else{
    gotoxy(20,2);
    cout << "Matricula inexistente";
    getch();
    pAnt = NULL;
    pAux = NULL;
  }
}


/******************* FUNCAO PRINCIPAL ******************/
void main(void){

   inicio.pProx = NULL; /* lista vazia */

   cabecalho();
   diario();
```

```
do{

 do{
   cabecalho();
   menu();
   if((int)opcao<0 || (int)opcao>3){
     opcao= -1;
     gotoxy(50,15);
     cout << "Opcao Invalida!";
   }
 }while(opcao == -1);
  switch(opcao){
   case 0:
        break;
   case 1:
        exibir();
        break;
   case 2:
        inserir();
        break;
   case 3:
        remover();
        break;
   default:
        printf("\nOpcao invalida");
        getch();
   }

 }while (opcao != 0);
}
```

Observação: o programa LS_INS02_dev.cpp é um exemplo mais avançado em relação ao programa LS_INS02.CPP que vai anexo a este material.

Exercício 2: Melhore o programa LS_INS02_dev.cpp para e amplie as funcionalidades conforme descrição a seguir.

a) Permitir que os dados sejam armazenas em secundária dos dados, bem como carregados desta para a memória principal.
b) Construção de relatório para ser visualizado na própria aplicação, ou mesmo que seja escrito em arquivo texto, a respeito de um aluno específico, e um outro relatório sobre informações a respeito de todos os alunos.
c) Retificação dos dados de um aluno conforme a matrícula fornecida.
d) Procure construir outras funções para que trabalhem com as telas (interfaces) e outras com as ações a serem realizadas, onde desta forma haverá uma separação melhor entre dados e telas.