Estruturas de Dados
Tipo Abstrato de Dados (TAD) Lista

Prof. Tales Nereu Bogoni
tales@unemat.br

UNEMAT
Universidade do Estado de Mato Grosso

Tipos Abstratos de Dados (TAD)

UNEMAT
Universidade do Estado de Mato Grosso

- Modelo matemático que encapsula um modelo de dados e um conjunto de procedimentos que atuam com exclusividade sobre os dados encapsulados.
- Em nível de abstração mais baixo, associado à implementação, esses procedimentos são implementados por subprogramas denominados operações, métodos ou serviços.
- Nesses casos, um programa baseado em TAD deverá conter algoritmos e estruturas de dados que implementem, em termos da linguagem de programação adotada, os procedimentos e os modelos de dados dos TADs utilizados pelo programa.
- a implementação de cada TAD pode ocupar porções bem definidas do programa: uma para a definição das estruturas de dados e outra para a definição do conjunto de algoritmos

Criação de uma estrutura de dados em C

UNEMAT
Universidade do Estado de Mato Grosso

- É utilizado o comando *struct*

```
struct estrutura {
    int a;
    int b;
    float c;
};

...

main()
{
    struct estrutura TAD;
{
```

- Para dar um nome à estrutura de dados utiliza-se o comando *typedef*

```
typedef struct {
    int a;
    int b;
    float c;
} Estrutura;
...
main()
{
    Estrutura TAD;
{
```

Atribuir valores

UNEMAT
Universidade do Estado de Mato Grosso

- Quando uma estrutura é criada, a atribuição é feita utilizando o nome da estrutura seguido do nome da variável (atributo), separado por ponto.

Estrutura TAD;

TAD.a=10;

TAD.b=30;

TAD.c=40.34;

Criação de bibliotecas

UNEMAT
Universidade do Estado de Mato Grosso

- Bibliotecas são locais aonde ficam armazenados os códigos fontes agrupados para uma finalidade específica
- Um TAD geralmente fica armazenado em uma biblioteca com sua estrutura de dados e com as funções que estão relacionadas a ele
- Possui dois arquivos
  - Arquivo ponto h
    - Tem as estruturas de dados
    - Tem os cabeçalhos (assinatura) as funções
    - Tem a importação de outras bibliotecas
  - Arquivo ponto c (ou cpp)
    - Tem a implementação das funções

Criação de um arquivo ponto h

UNEMAT
Universidade do Estado de Mato Grosso

- Deve ser carregado apenas uma vez, por isso é necessário fazer uma validação para ver se ainda não foi carregado pelo programa

```
#ifndef LISTA_H_INCLUDED
#define LISTA_H_INCLUDED
...
Implementação
...
#endif // LISTA_H_INCLUDED
```

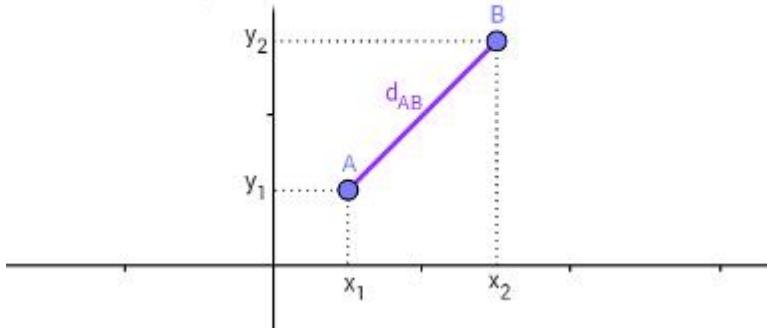
- Caso seja necessário devem ser inseridas as bibliotecas que serão usadas

```
#ifndef LISTA_H_INCLUDED
#define LISTA_H_INCLUDED
#include <stdlib.h>
...
Implementação
...
#endif // LISTA_H_INCLUDED
```

Exemplo - Criar uma biblioteca para realizar operações com Pontos no plano cartesiano (x e y)

- Quais os dados são necessários para representar um ponto?
- Uma das funções que se deseja é saber a distância euclidiana entre 2 pontos.

$$d_{AB} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$




```
#ifndef PONTO_H_INCLUDED
#define PONTO_H_INCLUDED

#include <math.h>

typedef struct {
    int x;
    int y;
} Ponto;

//Cria um ponto
void setPonto(Ponto *p, int x, int y);
//Calcula a distância entre dois pontos
float distanciaPonto(Ponto p1, Ponto p2);

#endif // PONTO_H_INCLUDED
```

Criação do arquivo ponto C

UNEMAT
Universidade do Estado de Mato Grosso

- Importar a biblioteca ponto H
- Implementa as funcionalidades

```
#include "Ponto.h"
void setPonto(Ponto *p, int x, int y)
{
   p->x=x;
   p->y=y;
}
float distanciaPonto(Ponto p1, Ponto p2)
{
   float distancia = sqrt( pow (p2.x - p1.x, 2) + pow (p2.y - p2.y, 2));
   return distancia;
}
```

Como utilizar a biblioteca?

UNEMAT
Universidade do Estado de Mato Grosso

- Importar o arquivo ponto H
  - Deve estar entre aspas

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

int main()
{
   Ponto p1,p2;
   setPonto(&p1,10,30);
   setPonto(&p2,40,20);
   float distancia = distanciaPonto(p1,p2);
   printf("A distancia dos pontos e %.4f/n",distancia);
}
```

Criação de um TAD para listas usando vetor

UNEMAT
Universidade do Estado de Mato Grosso

- Informações necessárias para criar uma lista

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

#define TAMANHO 20
int posicaoatual=0;
int lista[TAMANHO];

int main()
{

}
```

São as mesmas do programa utilizado anteriormente, porém dentro de uma estrutura

```
typedef struct{
    int posicaoatual;
    int tamanho;
    int *dados; // permite listas com tamanhos diferentes
} Lista;
```

Funções da lista

UNEMAT
Universidade do Estado de Mato Grosso

- Criar a lista
- Verificar se a lista está vazia
- Verificar se a lista está cheia
- Saber quantos elementos tem na lista
- Inserir um elemento no fim da lista
- Saber que elemento tem em uma determinada posição
- Pesquisar um elemento
- Remover um elemento do fim da lista
- Inserir um elemento no meio da lista
- Remover um elemento do meio da lista

Criar uma lista

UNEMAT
Universidade do Estado de Mato Grosso

- É necessário alocar a posição da memória pra a quantidade de elementos da lista
- Definir o tamanho da lista
- Setar a posição atual para 0

```
void criarLista(Lista *l, int tamanho)
{
    l->dados = malloc(tamanho*sizeof(int));
    l->tamanho = tamanho;
    l->posicaoatual=0;
}
```

Lista vazia ou cheia

UNEMAT
Universidade do Estado de Mato Grosso

```
int vazia(Lista *l)
{
   return (0 == l->posicaoatual);
}
```

```
int cheia(Lista *l)
{
   return (l->tamanho == l->posicaoatual);
}
```

Quantos elementos tem a lista?

UNEMAT
Universidade do Estado de Mato Grosso

```
int elementos(Lista *l)
{
   return l->posicaoatual;
}
```

Inserir elementos no final da lista

UNEMAT
Universidade do Estado de Mato Grosso

- Verificar se a lista não está cheia
- Inserir o elemento na posição atual da lista
- Incrementar a posição atual da lista
- Se inserir retorna um valor verdadeiro
- Se não inserir retorna um valor falso

```
int insere(Lista *l, int elemento)
{
   if(!cheia(l))
   {
      l->dados[l->posicaoatual]=elemento;
      l->posicaoatual++;
      return 1;
   }
   return 0;
}
```

Remover o elemento do final da lista

UNEMAT
Universidade do Estado de Mato Grosso

- Verificar se a lista não está vazia
- Decrementar a posição atual da lista
- Se a lista possuir elementos devolver o elemento e informar que realizou com sucesso operação *(true)*
- Se a lista estiver vazia retornar um valor *nulo* e dizer que não foi possível realizar a operação *(false)*

```
int removeUltimo(Lista *l, int *elemento)
{
   if(!vazia(l))
   {
      *elemento=l->dados[l->posicaoatual-1];
      l->posicaoatual--;
      return 1;
   }
   return 0;
}
```

Exibir um elemento da lista

UNEMAT
Universidade do Estado de Mato Grosso

- Passar a posição do elemento que deve ser exibido
- Caso exista, devolver o valor do elemento e informar que foi possível realizar a operação *(true)*
- Caso não exista, informar que não possível realizar a operação *(false)*

```
int elemento(Lista *l, int posicao, int *elemento)
{
   if(posicao>=0 && posicao<l->posicaoatual)
   {
      *elemento=l->dados[posicao];
      return 1;
   }
   return 0;
}
```

Pesquisar um elemento na lista

UNEMAT
Universidade do Estado de Mato Grosso

- Receber o valor que será pesquisado
- Percorrer a lista para ver se o elemento existe
- Se existe, retornar a posição dele e dizer que a operação foi concluída com sucesso *(true)*
- Se não existe, informar que não possível realizar a operação *(false)*

```
int perquisar(Lista *l, int valor, int*posicao)
{
   if(vazia(l))
      return 0;
   for(int i=0;i<l->posicaoatual;i++)
   {
      if(l->dados[i]==valor)
      {
         *posicao=i;
         return 1;
      }
   }
   return 0;
}
```

Inserir elementos no meio da lista

UNEMAT
Universidade do Estado de Mato Grosso

- Passar a posição e o valor do elemento que será inserido
- Se a lista está vazia, inserir o elemento no final
- Se a lista está cheia, não inserir o elemento
- Todos os elementos a partir da posição de inserção devem ser rearranjados na lista
- Informar se a inserção foi realizada com sucesso ou não

```
int insereMeio(Lista *l, int posicao, int valor)
{
   if(cheia(l) || posicao<0)
      return 0;
   if(vazia(l))
      return insere(l, valor);
   if(posicao>=l->posicaoatual)
      return insere(l, valor);
   for(int i=l->posicaoatual;i!=posicao;i--)
   {
      l->dados[i]=l->dados[i-1];
   }
   l->dados[posicao]=valor;
   l->posicaoatual++;
   return 1;
}
```

Remover elementos do meio da lista

UNEMAT
Universidade do Estado de Mato Grosso

- Verificar se a lista não está vazia
- Verificar se o elemento está dentro do vetor
- Realocar os elementos para ocupar a posição removida
- Revolver o elemento removido
- Informar se a remoção foi realizada com sucesso ou não

```
int removeMeio(Lista *l, int posicao, int*valor)
{
   if(vazia(l) || posicao<0 || posicao>=l->posicaoatual)
      return 0;
   if(elemento(l, posicao,valor))
   {
      for(int i=posicao;i<l->posicaoatual;i++)
         l->dados[i]=l->dados[i+1];
      l->posicaoatual--;
      return 1;
   }
   return 0;
}
```

Como usar?

UNEMAT
Universidade do Estado de Mato Grosso

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

int main()
{
Lista lista;
   Lista lista1;
   criarLista(&lista,10);
   criarLista(&lista1,5);
   if(insere(&lista,10))
      printf("Inserido\n");
   if(insere(&lista,20))
      printf("Inserido\n");
   if(insere(&lista1,1))
      printf("Inserido\n");
```

```
if(insereMeio(&lista,1,25))
      printf("Inserido\n");

   int v;
   if(removeUltimo(&lista,&v))
      printf("removido %d\n",v);

   printf("Lista \n");
   for(int i=0;i<lista.posicaoatual;i++)
      printf("%d - %d\n",i,lista.dados[i]);

   printf("Lista1 \n");
   for(int i=0;i<lista1.posicaoatual;i++)
      printf("%d - %d\n",i,lista1.dados[i]);
   return 1;
}
```

Exercício

UNEMAT
Universidade do Estado de Mato Grosso

- Criar duas listas, uma com 20 e outra com 40 elementos
- Povoar a lista com 20 elementos até enche-la
- Remover todos os elementos da lista 1 e inseri-los na lista 2
- Inserir 10 elementos na lista 1
- Inserir 5 elementos no meio da lista 2
- Pesquisar elementos aleatórios na lista 2 até encontrar um
- Excluir este elemento
- Apague todos os elementos das listas
- Mostrar todos os elementos da listas a cada tarefa
- Criar uma função para concatenar duas listas