Estruturas de Dados I
Lista utilizando vetores

Prof. Tales Nereu Bogoni
tales@unemat.br

UNEMAT
Universidade do Estado de Mato Grosso

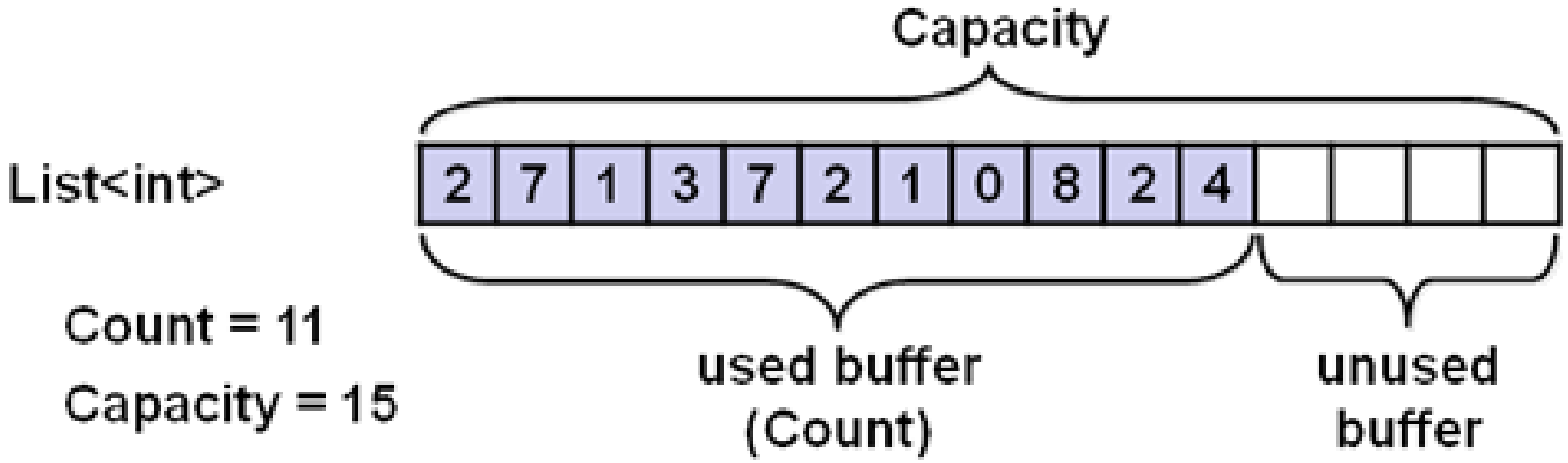
Listas

UNEMAT
Universidade do Estado de Mato Grosso

- Listas são estruturas extremamente flexíveis que possibilitam uma ampla manipulação das informações uma vez que inserções e remoções podem acontecer em qualquer posição
- As listas são estruturas compostas, constituídas por dados de forma a preservar a relação de ordem linear entre eles
- Podem ser de qualquer tipo de dado
- Em geral, uma lista segue a forma a1, a2, a3, ..., an
  - onde n determina o tamanho da lista
- Quando n = 0 a lista é chamada nula ou vazia.
- Para toda lista, exceto a nula, a_{i+1} segue (ou sucede) a_i (i < n), e a_{i-1} precede a_i (i > 1)
- O primeiro elemento da lista é a_1, e o último a_n

Definição de Lista

- Em geral, uma lista segue a forma a1, a2, a3, ..., an
  - onde n determina o tamanho da lista
- Quando n = 0 a lista é chamada nula ou vazia.
- Para toda lista, exceto a nula, a_{i+1} segue (ou sucede) a_i (i < n), e a_{i-1} precede a_i (i > 1)
- O primeiro elemento da lista é a_1, e o último a_n
- A posição correspondente ao elemento a_i na lista é *i*



Características

UNEMAT
Universidade do Estado de Mato Grosso

- Homogênea. Todos os elementos da lista são do mesmo tipo
- A ordem nos elementos é decorrente da sua estrutura linear, no entanto os elementos não estão ordenados pelo seu conteúdo, mas pela posição ocupada a partir da sua inserção
- Para cada elemento existe anterior e seguinte, exceto o primeiro, que não possui anterior, e o último, que não possui seguinte
- É possível acessar e consultar qualquer elemento na lista
- É possível inserir e remover elementos em qualquer posição

Operações Básicas com Listas

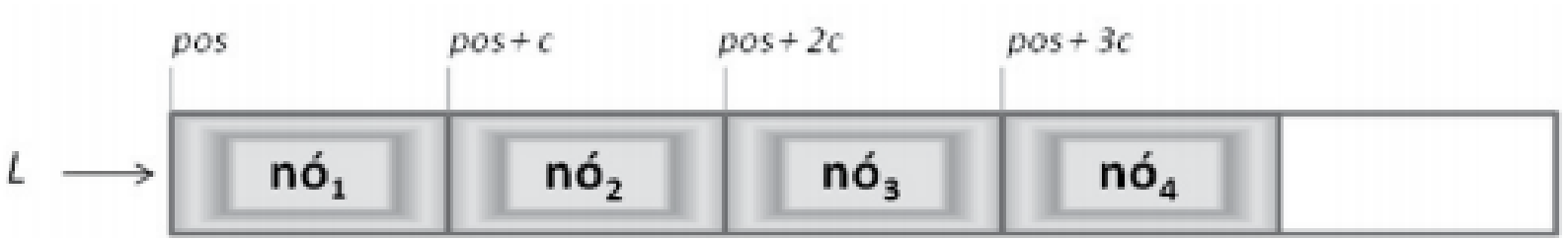
UNEMAT
Universidade do Estado de Mato Grosso

- Criar a lista
- Verificar se a lista está cheia
- Verificar se a lista está vazia
- Saber a quantidade de elementos de uma lista
- Inserir elementos no final da lista
- Remover um elemento no fim da lista
- Exibir um elemento da lista
- Pesquisar um elemento
- Inserir elementos no meio da lista
- Remover um elemento do meio da lista

Implementação usando alocação estática de memória

UNEMAT
Universidade do Estado de Mato Grosso

- Na implementação de lista adotando alocação de memória estática os elementos componentes são organizados em posições contíguas de memória utilizando vetores
- É necessário saber o tamanho do vetor que receberá a lista, este valor será a a quantidade máxima de elementos que ela poderá armazenar
- Nesta etapa definiremos um tipo de dados primitivo



Criação da lista

UNEMAT
Universidade do Estado de Mato Grosso

- Nesta etapa ainda não estamos trabalhando do TAD

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

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

int main()
{

}
```

Considerando que a posição atual é o número de elementos da lista e o TAMANHO é capacidade máxima de armazenamento.

Como saber se a lista está vazia?

posicaoatual == 0

Como saber se a lista está cheia?

posicaoatual == TAMANHO

Lista vazia?

UNEMAT
Universidade do Estado de Mato Grosso

```
int vazia()
{
    if(0 == posicaoatual)
        return 1;
    else
        return 0;
}
```

```
int vazia()
{
    return (0 == posicaoatual);
}
```

Lista Cheia?

UNEMAT
Universidade do Estado de Mato Grosso

```
int cheia()
{
    if(TAMANHO == posicaoatual)
        return 1;
    else
        return 0;
}
```

```
int cheia()
{
    return (TAMANHO == posicaoatual);
}
```

Quantos elementos tem a lista?

UNEMAT
Universidade do Estado de Mato Grosso

```
int tamanho()
{
    return 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(int elemento)
{
   if(!cheia())
   {
      lista[posicaoatual]=elemento;
      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)*

```
bool removeUltimo(int *elemento)
{
   if(!vazia())
   {
      *elemento=lista[posicaoatual-1];
      posicaoatual--;
      return true;
   }
   return false;
}
```

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 getelemento(int posicao, int *elemento)
{
   if(!vazia() && posicao<posicaoatual)
   {
      *elemento=lista[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)*

```
bool pesquisar(int valor, int *posicao)
{
   if(vazia())
      return false;
   for(int i=0;i<posicaoatual;i++)
   {
      if(lista[i]==valor)
      {
         *posicao=i;
         return true;
      }
   }
   return false;
}
```

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

```
bool insereMeio(int posicao, int valor)
{
   if(cheia())
      return false;
   if(posicao<0)
      return false;
   if(vazia())
      return insere(valor);
   if(posicao>=posicaoatual)
      return insere(valor);
   for(int i=posicaoatual;i!=posicao;i--)
         lista[i]=lista[i-1];
   lista[posicao]=valor;
   posicaoatual++;
   return true;
}
```

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
- Devolver o elemento removido
- Informar se a remoção foi realizada com sucesso ou não

```
bool removeMeio(int posicao, int*valor)
{
   if(vazia() || posicao<0 || posicao>=posicaoatual)
      return false;
   if(getelemento(posicao,valor))
   {
      for(int i=posicao;i<posicaoatual;i++)
         lista[i]=lista[i+1];
      posicaoatual--;
      return true;
   }
   return false;
}
```

Como usar?

UNEMAT
Universidade do Estado de Mato Grosso

```
int main()
{
   int valor;
   printf("%s\n",(vazia()==1?"Lista Vazia":"Lista com
Elementos"));
   if(insere(10))
     printf("Elemento inserido\n");
   if(insereMeio(0,5))
      printf("Elemento inserido\n");
   printf("A lista tem %d elementos\n",elementos());
   for(int i=0;i<posicaoatual;i++)
   {
      int temelemento = elemento(i,&valor);
      if(temelemento)
         printf("%d - %d\n",i,valor);
   }
   int deletado = removeUltimo(&valor);
   if(deletado)
      printf("Elemento removido %d\n",valor);
 if(insere(20))
     printf("Elemento inserido\n");
   if(insere(30))
     printf("Elemento inserido\n");
   int p;
   if(pesquisar(20,&p))
      printf("Encontrou o elemento na posicao %d\n",p);
   else
      printf("Elemento não encontrado\n");
   deletado = removeMeio(p,&valor);
   if(deletado)
      printf("Elemento %d removido da posicao
%d\n",valor,p);

   for(int i=0;i<posicaoatual;i++)
   {
      int temelemento = elemento(i,&valor);
      if(temelemento)
         printf("%d - %d\n",i,valor);
   }
   return 1;
}
```

Exercício

UNEMAT
Universidade do Estado de Mato Grosso

- Criar uma lista com até 50 elementos
- Povoar a lista com 40 elementos aleatórios
- Inserir outros 5 elementos aleatórios no meio da lista
- Pesquisar 10 elementos aleatórios e mostrar suas posições ou se eles não existem
- Remover os 3 últimos elementos da lista e mostrar seus valores
- Pesquisar elementos aleatórios na lista até encontrar um
- Excluir este elemento
- Inserir mais 15 elementos e informar se conseguiu ou não fazer a inclusão de cada um
- Apague todos os elementos da lista
- Mostrar todos os elementos da lista a cada tarefa