**Lista Duplamente Encadeada Dinâmica**

|  |
| --- |
| **ALGORITMO .::. CriarLDE** |
| *Entrada*: --  *Saída*: Ptlista (Tipo Nó); |
| **Início**  Alocar(Ptlista)  Se Ptlista = nulo então  Escrever(Erro na solicitação de memória)  Ptlista↑.ant ← nulo  ptlista↑.prox ← nulo  **fim** |

|  |
| --- |
| **ALGORITMO .::. InserirDepoisLDE** |
| *Entrada*: Ptlista, Ptant (tipo Nó) dado (tipo dado)  *Saída*: Ptlista (Tipo Nó);  *Variáveis auxiliares*: Novo, Aux (Tipo Nó) |
| **Início**  Alocar(novo)  Novo↑.info ← dado  Aux ← Ptant↑.prox  **Se** Aux ≠ nulo **então**  Aux↑.ant = Novo  Novo↑.prox ← Aux  Ptant↑.prox ← Novo  Novo↑.ant ←Ptant  **fim** |

|  |
| --- |
| **ALGORITMO .::. InserirPosicaoLDE** |
| *Entrada*: Ptlista (tipo Nó); k (inteiro); dado (tipo dado)  *Saída*: Ptlista (Tipo Nó);  *Variáveis auxiliares*: Ant, Aux (Tipo Nó); i (inteiro) |
| **Início**  **Se** k < 1 **Então**  **Início**  Escrever(posição inválida)  **Retorne**  **Fim**  **Senão**  **Início**  i ← 1  Aux ← Ptlista↑.prox  Ant ← Ptlista  **Enquanto** (i < k) E (aux ≠ nulo) **então**  **Início**  Ant ← Aux  Aux ← Aux↑.prox  i ← i + 1  **Fim**  **Se** K > i +1 **Então**  **Início**  Escrever(POSIÇÃO INVÁLIDA)  **Retorne**  **Fim**  **Senão**  {AlGORITMO inserirDepois\_LDE(Ptlista, Ant, Dado)}  **Fim** |

|  |
| --- |
| **ALGORITMO .::. RemoverNó\_LDE** |
| *Entrada*: Ptlista, P (tipo Nó)  *Saída*: Ptlista (Tipo Nó);  *Variáveis auxiliares*: R, Q (Tipo Nó) |
| **Início**  **Se (**p = nulo) ou (Ptlista↑.prox = nulo) **Então**  Escrever(Lista vazia)  **Senão**  **Início**  Q ← P↑.prox  R ← P↑.ant  R↑.prox ← P↑.prox  **Se** Q ≠ nulo **então**  Q↑.ant ← P↑.ant  Desalocar(P)  **Fim**  **fim** |

|  |
| --- |
| **ALGORITMO .::. ExcluirPosicao\_LDE** |
| *Entrada*: Ptlista,(tipo Nó) ; k (inteiro)  *Saída*: Ptlista (Tipo Nó);  *Variável auxiliar*: i (inteiro); Atual(Tipo Nó) |
| **Início**  **Se** Ptlista↑.prox = nulo **Então**  Escrever(Lista vazia)  **Senão**  **Início**  Atual ← Ptlista↑.prox  **Para** i de 1 ate k – 1 **faça**  Atual ←Atual↑.prox  {Algoritmo removerNó\_LDE(Ptlista, Atual)}  **Fim**  **fim** |

|  |
| --- |
| **ALGORITMO .::. PesquisarPorElemento\_LDE** |
| *Entrada*: Ptlista (tipo Nó), dado(tipo dado)  *Saída*: Item (Tipo Nó) |
| **Início**  Item ←Ptlista↑.prox  **Enquanto** item ≠ nulo **Faça**  **Inicio**  **Se** dado = item↑.prox **Então**  Parar Iteração  **Senão**  Item ← item↑.prox  **Fim**  **Fim** |