A teoria sobre listas lineares

Lista Linear

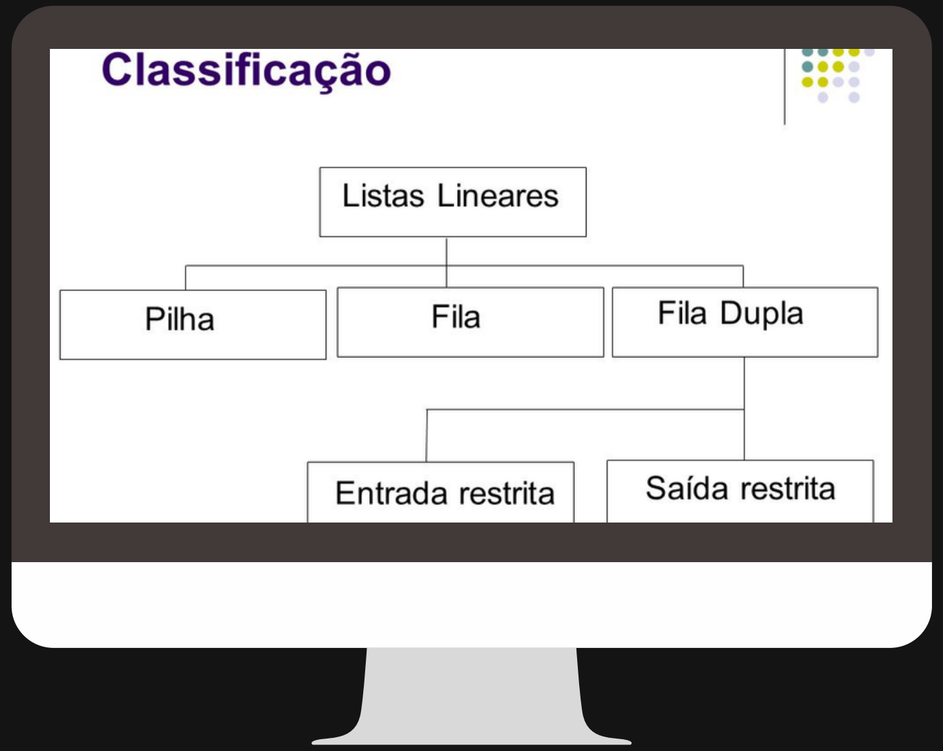
Uma lista linear é uma estrutura de dados na qual cada elemento da lista é A frente é um elemento e a parte de trás é outro elemento. mas não isso acontece no primeiro elemento porque não tem predecessor, e O último não tem sucessor.

Lista Linear Sequencial

A partir da lista, os itens são organizados em ordem sequencial. As imagens são armazenadas em locais de memória que não são parados. É importante entender quantos elementos diferentes a lista contém. Antes de realizar qualquer operação, os dados são armazenados primeiro.

Tipos de Listas Lineares

- Pilha
- Filas
- Deque ou filas duplas



Lista Linear Sequencial

A partir da lista, os itens são organizados em ordem sequencial. As imagens são armazenadas em locais de memória que não são parados. É importante entender quantos elementos diferentes a lista contém. Antes de realizar qualquer operação, os dados são armazenados primeiro.

Hora do código

```
struct  Node{
 int  num;
 struct  Node  *prox;
};
typedef  struct   Node  node ;

int  tam;

void  cabecalho ();
void  inicia(node  *LISTA);
int  menu(void);
void  opcao(node  *LISTA, int  op);
node  *criaNo();
void  insereFim(node  *LISTA);
void  inserelnicio (node  *LISTA);
void  exibe(node  *LISTA);
void  libera(node  *LISTA);
node  *retiralnicio (node  *LISTA);
node  *retiraFim(node  *LISTA);
node  *retira(node  *LISTA);
```

```c
int  main(void)
{
 node  *LISTA  =  (node  *)  malloc(sizeof( node));
 if(! LISTA){
  printf ("Sem memoria disponivel!\n"  );
  exit (1);
 }else{
 inicia(LISTA);
 int  opt;

 do{
  opt =menu();
  opcao(LISTA,opt);
 }while( opt);

 free(LISTA);
 return  0;
 }
}
```

```c
void inicia(node *LISTA)
{
 LISTA->prox = NULL;
 tam=0;
}

int menu(void)
{
 int opt;
 printf("Escolha a opcao\n");
 printf("1. Adicionar numero no inicio\n");
 printf("2. Adicionar numero no final\n");
 printf("3. Exibir lista\n");
 printf("4. Retirar numero do inicio\n");
 printf("5. Retirar numero do fim\n");
 printf("6. Sair\n");
```

```
void opcao(node *LISTA, int op)
{
 node *tmp;
 cabecalho();
 switch(op){
  case 6:
   libera(LISTA);
   break;


  case 3:
   exibe(LISTA);
   break;

  case 1:
   insereInicio(LISTA);
   break;

  case 2:
   insereFim(LISTA);
   break;


  case 4:
   tmp= retiraInicio(LISTA);
   printf("Retirado: %3d\n\n", tmp->num);
   break;

  case 5:
   tmp= retiraFim(LISTA);
   printf("Retirado: %3d\n\n", tmp->num);
   break;

  default:
   printf("Comando invalido\n\n");
 }
}
```

```c
int vazia(node *LISTA)
{
  if(LISTA->prox == NULL)
    return 1;
  else
    return 0;
}

node *aloca()
{
  node *novo=(node *) malloc(sizeof(node));
  if(!novo){
    printf("Sem memoria disponivel!\n" );
    exit(1);
  }else{
    printf("Novo numero: " ); scanf("%d", &novo->num);
    return novo;
  }
}
```

```c
void insereFim(node *LISTA)
{
 node *novo=aloca();
 novo->prox = NULL;

 if(vazia(LISTA))
  LISTA->prox=novo;
 else{
  node *tmp = LISTA->prox;

  while(tmp->prox != NULL)
   tmp = tmp->prox;

  tmp->prox = novo;
 }
 tam++;
}

void inserelnicio (node *LISTA)
{
 node *novo=aloca();
 node *oldHead = LISTA->prox;

 LISTA->prox = novo;
 novo->prox = oldHead;

 tam++;
}
```

```c
void exibe(node *LISTA)
{
 system("clear");
 if(vazia(LISTA)){
  printf("Lista vazia!\n\n");
  return;
 }

 node *tmp;
 tmp = LISTA->prox;
 printf("Lista:");
 while(tmp != NULL){
  printf("%5d", tmp->num);
  tmp = tmp->prox;
 }
 printf("\n       ");
 int count;
 for(count=0; count < tam; count++)
  printf("  ^  ");
 printf("\nOrdem:");
 for(count=0; count < tam; count++)
  printf("%5d", count+1);


 printf("\n\n");
}
```

```c
void libera(node *LISTA)
{
 if(!vazia(LISTA)){
  node *proxNode,
       *atual;

  atual = LISTA->prox;
  while(atual != NULL){
   proxNode = atual->prox;
   free(atual);
   atual = proxNode;
  }
 }
}

node *retiralnicio(node *LISTA)
{
 if(LISTA->prox == NULL){
  printf("Lista ja esta vazia\n");
  return NULL;
 }else{
  node *tmp = LISTA->prox;
  LISTA->prox = tmp->prox;
  tam--;
  return tmp;
 }

}
```

```
node *retiraFim(node *LISTA)
{
 if(LISTA->prox == NULL){
  printf("Lista ja vazia\n\n");
  return NULL;
 }else{
  node *ultimo = LISTA->prox,
     *penultimo = LISTA;

  while(ultimo->prox != NULL){
   penultimo = ultimo;
   ultimo = ultimo->prox;
  }

  penultimo->prox = NULL;
  tam--;
  return ultimo;
 }
}
void cabecalho()
{
     system("cls");
}
```